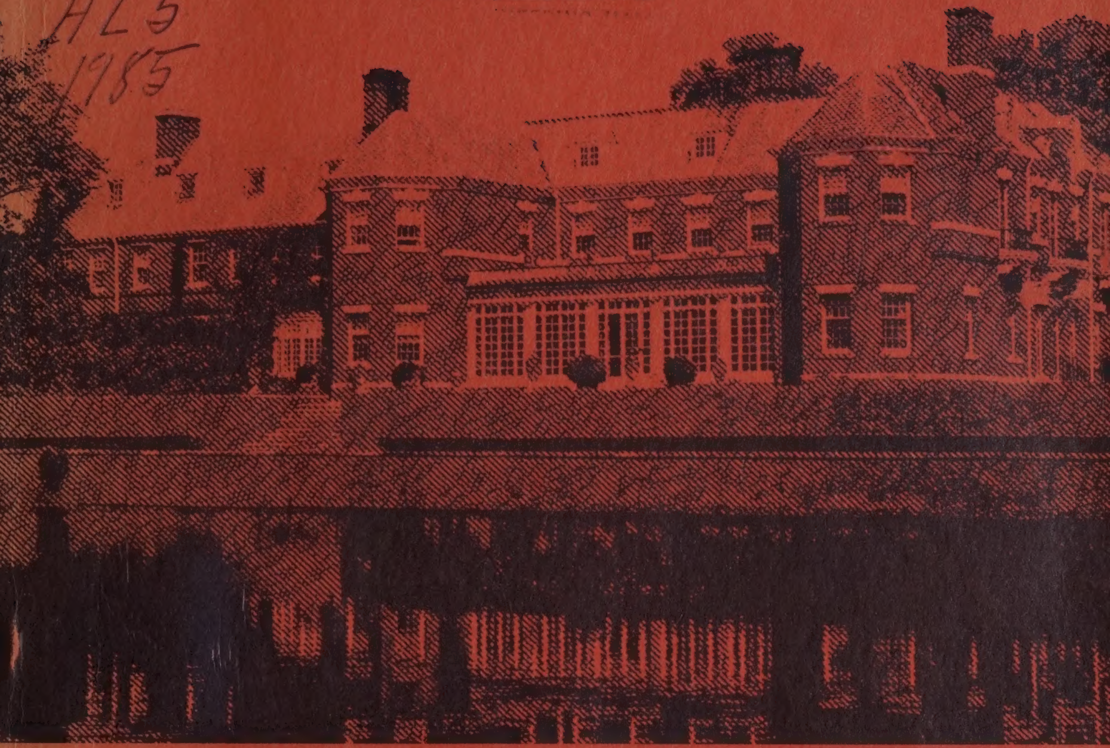
621.3811
AL5
1985

ENGINEERING LIBRARY



Proceedings

TWENTY-THIRD ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

October 2-4, 1985

Allerton House, Monticello, Illinois
Sponsored by the
Coordinated Science Laboratory and the
Department of Electrical and Computer Engineering of the
University of Illinois at Urbana-Champaign

NOTICE: **Return or renew all Library Materials! The *Minimum* Fee for each Lost Book is $50.00.**

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.
To renew call Telephone Center, 333-8400

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

ENGINEERING

PHOTO REPRODUCTION

AUG 05 1992

JUL 16 REC'D

JAN 03 1994

DEC 06 REC'D

L161—O-1096

PROCEEDINGS

TWENTY-THIRD ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL, AND COMPUTING

Bruce Hajek
David C. Munson, Jr.
Conference Co-Chairmen

Conference held
October 2-4, 1985
Allerton House
Monticello, Illinois

sponsored by
The Coordinated Science Laboratory
and
The Department of Electrical and Computer Engineering
of the
UNIVERSITY OF ILLINOIS
at
Urbana-Champaign

FOREWORD

The Twenty-Third Annual Allerton Conference on Communication, Control and Computing was held at Allerton House, Monticello, Illinois on October 2, 3, and 4, 1985. The conference was sponsored by the Coordinated Science Laboratory and the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. These *Proceedings* contain the papers presented at the Conference.

We are grateful to Professor Richard M. Karp of the University of California at Berkeley for his enchanting plenary lecture entitled, "On the Complexity of Parallel Computation." We are also very grateful to Professor P.R. Kumar of the University of Illinois for organizing the invited session, "Recent Advances in Systems and Control: Identification and Adaptive Control," and to Prof. S. Verdu of Princeton University for organizing the invited session, "Optical Communication." The first of these two invited sessions was supported by the Mathematics Steering Committee of the U.S. Army Research Office. We would like to thank all the session chairmen and the authors for their fine contributions.

We appreciate the support of Professor M. E. VanValkenburg, Dean of the College of Engineering and founder of the series of Allerton Conferences, Professor T. N. Trick, Director of the Coordinated Science Laboratory and Head of the Department of Electrical and Computer Engineering, and Professor J. B. Cruz, Jr., Associate Director of the Coordinated Science Laboratory and Associate Head of the Department of Electrical and Computer Engineering. We would also like to thank our colleagues, Professors J. A. Abraham, K. S. Arun, P. Banerjee, T. M. Basar, T. U. Basar, J. Grizzle, W. K. Fuchs, I. N. Hajj, T. S. Huang, M. Ilic-Spong, R. Iyer, W. K. Jenkins, P. V. Kokotovic, P. R. Kumar, H. Lee, C. L. Liu, M. C. Loui, J. Medanic, J. H. Patel, W. R. Perkins, K. R. Poola, H. V. Poor, F. P. Preparata, M. B. Pursley, V. Ramachandran, D. V. Sarwate, and M. W. Spong for their generous assistance in reviewing papers (over 260 were submitted) and organizing the papers into sessions.

We are grateful to Mr. Steven Williams for his handling of the business affairs of the Conference, to Mr. Robert Ebeling and his assistants for their timely printing of large numbers of programs and registration packets, and to Ms. Rose Harris, Ms. Tina Lipscomb, Ms. Rosemary Wegeng and Ms. Martha Watkins for their capable assistance in handling the Conference announcement, registration, program preparation, and editing and sales of these *Proceedings*.

Thanks are due to Dr. E. Dalheim, Ms. B. Hedlund and Dr. H. Klug for their outstanding musical presentation. Finally, we offer our sincere thanks to Franco Preparata, whose vigilance for the slightest impropriety constantly kept us on our toes.

Bruce Hajek
David C. Munson, Jr.

TABLE OF CONTENTS

FOREWORD ... iii

PLENARY LECTURE

THE COMPLEXITY OF PARALLEL COMPUTATION ... 1
R. M. Karp

I-A: ESTIMATION AND DETECTION

Chairman: W.L. Root
University of Michigan

EXACT NONLINEAR FILTERS, EXPONENTIAL DENSITIES AND SEPARATION OF VARIABLES ... 2
F. E. Daum

FINITE DIMENSIONAL FILTERS FOR A CLASS OF NONLINEAR SYSTEMS THAT ARE IMMERSIBLE IN A LINEAR SYSTEM ... 9
J. Levine

A MINIMAX FILTERING THEORY FOR COLORED NOISE ... 18
R. K. Bahr and J. A. Bucklew

A WELL-POSEDNESS PROPERTY OF A CLASS OF VARIATIONAL PROBLEMS AND ITS APPLICATION TO NONLINEAR ESTIMATION ... 28
J. M. Morrison and G. L. Wise

SIGNAL-TO-NOISE RATIO AND CENTRAL LIMIT THEOREM CONSIDERATIONS IN NON-GAUSSIAN DETECTION ... 35
D. Warren and J. B. Thomas

ASYMPTOTICALLY ROBUST DETECTION OF STOCHASTIC SIGNALS IN CONTAMINATED NOISE ... 45
M. S. Schnitzer and D. R. Halverson

A LEAST FAVORABLE DEPENDENCY STRUCTURE FOR LARGE SIGNAL DETECTION PERFORMANCE ... 53
J. S. Sadowsky

I-B: COMBINATORIAL ALGORITHMS

Chairman: E. Reingold
University of Illinois

ON THE STOCHASTIC SCHEDULING OF A TREE ... 63
C. H. Papadimitriou and J. N. Tsitsiklis

LAYER STRUCTURE IN SERIES-PARALLEL GRAPHS 67
M. M. Kouta and S. E. Conry

ANOMALOUS BEHAVIOR IN BIN PACKING ALGORITHMS 74
F. D. Murgolo

A FAST LINEAR SPACE ALGORITHM FOR COMPUTING LONGEST COMMON SUBSEQUENCES 76
A. Apostolico and C. Guerra

A VERY EFFICIENT ALGORITHM FOR FINDING ALL CYCLES IN A DIGRAPH 85
X. Jin

A LINEAR DIRECTED CUT TRANSVERSAL ALGORITHM FOR 1-SINK ACYCLIC GRAPHS 92
K. Vidyasankar

GENERALIZED K-LEVEL SEARCH REARRANGEMENT 101
K. M. Bugara and P. W. Purdom, Jr.

I-C: NONLINEAR SYSTEMS

Chairman: R. Marino
Italy

AN APPLICATION OF VOLTERRA SERIES TO THE DESIGN OF NONLINEAR FEEDBACK CONTROLS 103
S. A. Al-Baiyat and M. K. Sain

DESIGN OF A COMMON FEEDBACK CONTROLLER FOR A FAMILY OF NONLINEAR SECOND ORDER SYSTEMS 113
G. Gu and E. B. Lee

A NEW APPROACH TO THE NONINTERACTING CONTROL PROBLEM IN NONLINEAR SYSTEMS THEORY 123
M. Fliess

ADAPTIVE HIGH-GAIN FEEDBACK CONTROL 130
A. Saberi and H. Khalil

A LYAPUNOV APPROACH FOR THE DESIGN OF VARIABLE STRUCTURE SYSTEMS 140
H. Sira-Ramirez

ANALYZING INTERCONNECTED SYSTEMS USING THE DUAL GENERALIZED HESSENBERG REPRESENTATION (GHR) 142
D. K. Lindner and R. C. Montgomery

GIBBS VECTOR KINEMATICS AND DECOUPLED ATTITUDE MANEUVERS 144
T.A.W. Dwyer, III

ANALYSIS OF TRANSIENT OSCILLATIONS IN A NONLINEAR FEEDBACK SYSTEM 146
R. K. Miller and A. N. Michel

A THEOREM ON THE FEEDBACK EQUIVALENCE OF NONLINEAR SYSTEMS USING TENSOR ANALYSIS 156
J. A. O'Sullivan and M. K. Sain

I-D: DIGITAL TESTING

Chairman: J. Patel
University of Illinois

LOGICAL FAULT ANALYSIS AND DESIGN FOR TESTABILITY OF PROGRAMMABLE LOGIC ARRAYS 158
P. Bose

VECTOR SETS FOR VLSI TESTING 168
G. Seroussi and N. H. Bshouty

ON THE GENERATION OF TEST VECTORS FOR MULTIPLE FAULTS IN DIGITAL CIRCUITS 174
S. Chakravarty and H. B. Hunt III

ANALYSIS OF FAULT COVERAGE MASKING IN BUILT-IN SELF-TEST SCHEMES 183
N. Cotsapas and V. K. Agarwal

MICROPROCESSOR TESTING 193
Z. Tong, L. L. Kinney and R. Y. Kain

A TEST PATTERN GENERATOR WITH EMBEDDED COMPRESSED OUTPUT FOR BUILT-IN SELF-TEST 202
P. Sheen, R. Dandapani and K. K. Saluja

II-A: RECENT ADVANCES IN SYSTEMS AND CONTROL: IDENTIFICATION AND ADAPTIVE CONTROL (SPECIAL INVITED SESSION)

Chairman: P. R. Kumar
University of Illinois

SYSTEM IDENTIFICATION: AN EXPOSITORY SURVEY 204
L. Ljung

ADAPTIVE CONTROL THEORY: THE LAST FIVE YEARS 217
K. S. Narendra

II-B: PARALLEL AND DISTRIBUTED COMPUTATION

Chairman: J. Simon
University of Chicago

AREA-TIME COMPLEXITY OF RECTANGULAR MATRIX MULTIPLICATION 233
K. F. Pang and A. El Gamal

A MATRIX FACTORIZATION APPROACH TO THE PARALLEL SOLUTION OF FIRST-ORDER LINEAR RECURRENCES 243
G. G. L. Meyer and L. J. Podrazik

ON USING MULTIPLE INVERTED TREES FOR PARALLEL UPDATING OF GRAPH PROPERTIES 251
S. Pawagi and I. V. Ramakrishnan

A FAMILY OF MERGING NETWORKS DERIVED FROM THE BITONIC MERGER 261
G. Bilardi

SIMULATIONS BETWEEN MESH-CONNECTED PROCESSOR ARRAYS 268
M. J. Atallah

CONTINUOUS ROUTING AND BROADCASTING ON INTERCONNECTION NETWORKS 270
Y. Chang, S. Hambrusch and J. Simon

ON THE COST OF SENSE OF DIRECTION IN DISTRIBUTED NETWORKS 280
Z. Harpaz and S. Zaks

AN OPTIMAL SHOUT-ECHO ALGORITHM FOR SELECTION IN DISTRIBUTED SETS 283
J. M. Marberg and E. Gafni

II-C: IMAGE PROCESSING

Chairman: K. S. Arun
University of Illinois

AN APPROACH TO LANDMARK-BASED SHAPE RECOGNITION 292
N. Ansari and E. J. Delp

NON-LINEAR ALIGNMENT OF CONTOURS FOR SHAPE CLASSIFICATION 301
L. Gupta and M. D. Srinath

ADAPTIVE L-FILTERS 304
A. C. Bovik and A. Restrepo

TEXTURE BOUNDARY DETECTION BASED ON LONG CORRELATION MODEL ..314
R. L. Kashyap and K.-B. Eom

TREE SEARCHED CHAIN CODING FOR SUBPIXEL RECONSTRUCTION OF LINE DRAWINGS......324
R. Sriraman, J. Koplowitz and S. Mohan

ITERATIVE RECONSTRUCTION OF BANDLIMITED IMAGES FROM NON UNIFORMLY SPACED SAMPLES......326
K. D. Sauer and J. P. Allebach

A CORRESPONDENCE BETWEEN DELAY-DOPPLER RADAR-IMAGING AND RADIONUCLIDE IMAGING......336
D. L. Snyder

II-D: DATA COMPRESSION

Chairman: K. Yao
UCLA

RATE DISTORTION BOUNDS FOR QUOTIENT BASED DISTORTION FUNCTIONS WITH APPLICATION TO LPC SYSTEMS......337
A. Buzo, F. Kuhlmann and C. Rivera

FAST VECTOR QUANTIZERS......347
N. Moayeri, D. L. Neuhoff and W. E. Stark

LOW RATE DISTRIBUTED QUANTIZATION OF NOISY OBSERVATIONS......354
R. M. Gray, S. Boyd and T. Lookabaugh

PERFORMANCE OF ENTROPY-CONSTRAINED BLOCK TRANSFORM QUANTIZERS......359
N. Farvardin and F. Y. Lin

SOME FAMILIES OF ZERO-ERROR CODES FOR THE 2-USER BINARY ADDER CHANNEL WITH FEEDBACK......361
Z. Zhang, T. Berger and J. L. Massey

II-E: OPTICAL COMMUNICATION

Chairman and Organizer: S. Verdu
Princeton University

E-M ALGORITHMS FOR ESTIMATING PARAMETERS FROM SINGLE-MEMORY MARKOV POINT-PROCESSES HAVING A MULTIPLICATIVE INTENSITY......370
M. I. Miller, N. A. Karamanos and W. R. Bosch

ESTIMATION OF THE RATE OF A DISCRETE-TIME MULTIVARIATE POINT PROCESS......372
J. Gubner and P. Narayan

COHERENT-DETECTION OPTICAL COMMUNICATION SYSTEMS: A PROOF OF THE GAUSSIAN BIT ERROR RATE......382
S. Verdu

PERFORMANCE CHARACTERISTICS OF LASER DIODE COMMUNICATION SYSTEMS WITH APD RECEIVERS AND COLOR-CODED PPM SIGNALING 388
M. Bayoumi and F. Davidson

ERROR EXPONENTS FOR A DIRECT DETECTION OPTICAL CHANNEL 390
P. A. Humblet

MULTI-PULSE PPM AND A NEW LOOK AT CODING FOR DIRECT-DETECTION OPTICAL CHANNELS USING APD BASED RECEIVERS 401
M. A. Herro and L. Hu

II-F: ADAPTIVE SYSTEMS AND ROBOTICS

Chairman: J. Chandra
U.S. Army Research Office

ADAPTIVE IDENTIFICATION OF SYSTEMS WITH POLYNOMIAL DEPENDENCE ON UNKNOWN PARAMETERS 411
S. Dasgupta

EFFECTS OF VARIABLE LOADS ON GROSS MOTION OF ADAPTIVELY CONTROLLED MANIPULATOR 417
S. Zarei and A. J. Koivo

INVESTIGATIONS INTO THE ROBUSTNESS OF SELF-TUNING CONTROL 419
K. Liu and J. Gertler

ADAPTIVE FORCE-POSITION-VELOCITY CONTROL FOR ROBOTIC MANIPULATORS 421
A. J. Koivo

III-A: CODING

Chairman: W. E. Stark
University of Michigan

MULTIDIMENSIONAL CODED PSK SYSTEMS USING UNIT-MEMORY TRELLIS CODES 428
A. LaFanechere and D. J. Costello, Jr.

VITERBI DECODING BY SYSTOLIC ARRAY 430
C. Y. Chang and K. Yao

CONCURRENT VITERBI ALGORITHM ON A HYPERCUBE 440
F. Pollara

ERROR-CORRECTING LINE CODES 450
M. A. Herro and L. Hu

METHODS FOR PROTECTION OF COMPUTER MEMORIES 452
M. Blaum, R.M.F. Goodman and R. J. McEliece

THE STRUCTURE AND COMPLEXITY OF LINEAR SUM CODES 455
T. Fuja, C. Heegard and R. Goodman

A PRELIMINARY ANALYSIS OF THE HOPFIELD ASSOCIATIVE MEMORY 465
R. J. McEliece, E. C. Posner and E. R. Rodemich

III-B: COMPUTATIONAL GEOMETRY

Chairman: G. Fredrikson
Purdue University

FINDING MINIMAL NESTED POLYGONS 470
S. Suri and J. O'Rourke

A LINEAR-TIME HEURISTIC FOR MINIMUM WEIGHT TRIANGULATION OF CONVEX POLYGONS 480
A. Lingas

FINDING THE LARGEST RECTANGLE IN AN ORTHOGONAL POLYGON 486
M. McKenna, J. O'Rourke and S. Suri

EFFICIENT HIDDEN-LINE ELIMINATION BY CAPTURING WINDING INFORMATION 496
J. A. Dean and J.-R. Sack

A POSSIBLY LINEAR MINIMUM SPANNING CIRCLE ALGORITHM 506
B. J. Oommen

GROWING AND SHRINKING OPERATIONS ON A MONOTONE POLYGON 508
G. Roth

THE ALGEBRAIC COMPLEXITY OF SHORTEST PATHS IN POLYHEDRAL SPACES 510
C. Bajaj

A ONE-DIMENSIONAL SYSTOLIC ARRAY FOR THE LARGEST EMPTY RECTANGLE PROBLEM 518
F. Dehne

III-C: COMMUNICATION NETWORK ACCESS AND CONTROL

Chairman: P. Humblet
Massachusetts Institute of Technology

POISSON CONFLICT RESOLUTION WITH FEEDBACK BASED ON CONFLICT INTENSITY 525
N. Mehravari

TECHNIQUES FOR ASSESSING THE STABILITY OF MULTIPLE-ACCESS PROTOCOLS 527
L. I. Sennott

VARIABLE LENGTH CODING SCHEMES FOR COLLISION-FREE CHANNEL ACCESS 529
D.W. Browning and J. B. Thomas

AN ANALYSIS OF THE SILENT CHANNEL ACCESS METHOD 531
A. P. Nadkarni and K.S. Vastola

AN ANALYSIS OF A CSMA/CD COLLISION RESOLUTION SCHEME 533
Y.-C. Liu and G. L. Wise

THE EFFECT OF ACKNOWLEDGEMENTS ON THE PERFORMANCE OF DISTRIBUTED SPREAD SPECTRUM PACKET RADIO NETWORK 543
S. S. Lee and J. A.Silvester

AN ITERATIVE MODEL TO ANALYZE MULTIHOP PACKET RADIO NETWORKS 545
S.-L. Su and V.O.K. Li

AN OPTIMIZATION PROBLEM IN A MULTISERVER QUEUE 555
I. Viniotis and A. Ephremides

OPTIMAL ROUTING AND FLOW CONTROL OF A NETWORK OF PARALLEL PROCESSORS WITH INDIVIDUAL BUFFERS 564
A. D. Bovopoulos and A. A. Lazar

III-D: SYSTEM IDENTIFICATION AND SPECTRUM ESTIMATION

Chairman: J. Allebach
Purdue University

RETRIEVAL OF SINUSOIDAL SIGNALS BY ADAPTIVE NOTCH FILTERING 574
L. Pearlstein and B. Liu

IMPULSE RESPONSE IDENTIFICATION WITH DETERMINISTIC INPUTS USING NON-PARAMETRIC METHODS 584
U. K. Bhargava, R. L. Kashyap and D. M. Goodman

A NOVEL METHOD FOR SOLVING EXTREME EIGENVECTORS 594
Y. H. Hu and P.-K. Chou

ITERATIVE METHODS FOR SUBSPACE COMPUTATION 604
D. R. Fuhrmann

A FAST PRONY ALGORITHM 606
J.R. Cruz and R. J. Mulholland

A NEW IMPLEMENTATION OF THE CONJUGATE GRADIENT METHOD FOR NONLINEAR SYSTEMS 616
B. J. Sullivan and J. M. Nocedal

III-E: LINEAR SYSTEMS

Chairman: B. R. Barmish
University of Wisconsin

CONTROLLABILITY OF SYSTEMS WITH LARGE DELAYS....618
A.S.C. Sinha

NEAR-DISTURBANCE LOCALIZATION USING SECOND-ORDER MODES....620
J. M. Saniuk and I.B. Rhodes

APPLICATION OF THE MULTIVARIABLE NYQUIST CRITERION TO DISTRIBUTED SYSTEMS....622
D. W. Luse

ON CAUSAL INPUT-OUTPUT SYSTEMS....624
W. L. Root and D. Serakos

MINIMAL REALIZATIONS OF MATRIX FRACTION DESCRIPTIONS....634
S.-Y. Zhang

IV-A: GEOMETRY AND LAYOUT

Chairman: C. L. Liu
University of Illinois

ALGORITHMS FOR THE MINIMUM K-PARTITIONING OF RECTILINEAR POLYGONS....644
O. Günther

ON PARTITIONING RECTILINEAR POLYGONS INTO STAR-SHAPED POLYGONS....654
R. Liu and S. Ntafos

MINIMUM LENGTH AND "THICKEST-FIRST" RECTANGULAR PARTITIONS OF POLYGONS....664
C. Levcopoulos

AN ALGORITHM FOR THREE LAYER CHANNEL ROUTING USING RESTRICTED OVERLAP....674
M. L. Brady and D. J. Brown

A FAST ALGORITHM FOR SINGLE ROW ROUTING....676
S. Han and S. Sahni

PLACEMENT THROUGH PAIRWISE INTERCHANGE AMONG CONNECTED AND UNCONNECTED LOGIC MODULES....686
S. K. Nandy and L. M. Patnaik

IV-B: NETWORK QUEUEING, SCHEDULING AND COMPUTATION

Chairman: E. Gafni
UCLA

OPTIMAL HOP-BY-HOP FLOW CONTROL IN COMPUTER NETWORKS 688
Z. Rosberg and I. S. Gopal

MODELLING THE FLOW OF COALESCING DATA STREAMS THROUGH A PROCESSOR 693
V. Anantharam

CALCULATING EXIT TIMES FOR SERIES JACKSON NETWORKS I 703
W. A. Massey

ANOTHER LOOK AT INSENSITIVITY 708
J. Walrand

THE NUMERICAL SOLUTION OF LINEAR DIFFERENTIAL EQUATIONS ON PARALLEL PROCESSORS BY ASYNCHRONOUS RELAXATIONS 710
D. Mitra

SCHEDULING DATA TRANSFERS IN NETWORKS WITH PRE-EMPTION 720
H.-A. Choi and S. L. Hakimi

DISTRIBUTED OPTIMAL SAMPLING WITH ASYNCHRONOUS COMMUNICATION 730
P. R. S. Kumar and G. K. Young

IV-C: ALGORITHMS AND ARCHITECTURES FOR DIGITAL SIGNAL PROCESSING

Chairman: B. Liu
Princeton University

A NONCLASSICAL WRLS ALGORITHM 732
T. C. Luk and J. R. Deller, Jr.

A FIXED-POINT COMPUTATION FOR LINEAR PHASE LATTICE FILTERING 742
D. C. Farden and J. R. Bellegarda

SYNTHESIS OF MINIMUM ROUNDOFF NOISE IN 3-D STATE SPACE DIGITAL FILTERS 752
A. Zilouchian and R. L. Carroll

QUANTIZATION ERROR REDUCTION IN DIGITAL CONTROL SYSTEMS 762
Y. A. Wu

DOUBLY PIPELINED CORDIC ARRAY PROCESSOR FOR SOLVING TOEPLITZ SYSTEMS....764
T.-Y. Sung, Y.-H. Hu, and H.-J. Yu

THE DEVELOPMENT AND TESTING OF THREE DFT ALGORITHMS ON THE HP-64000 MICROPROCESSOR LOGIC DEVELOPMENT SYSTEM....774
P. D. Stigall, X. Li and D. F. Dawson

IV-D: STOCHASTIC SYSTEMS

Chairman: H. Khalil
Michigan State University

BILINEAR PROGRAMMING AND STRUCTURED STOCHASTIC GAMES....776
J. A. Filar and T. A. Schultz

PIECEWISE LINEAR MODELING OF MULTIDIMENSIONAL STOCHASTIC NONLINEAR SYSTEMS....777
A. H. Haddad and E. I. Verriest

SOME NEW RESULTS IN OVERLAPPING PARAMETRIZATION OF MULTIVARIABLE SYSTEMS....779
S. Bingulac and R. Krtolica

STABILITY PROPERTY OF THE CERTAINTY EQUIVALENT CONTROL OF DIGITAL RANDOM-PARAMETER SYSTEMS....781
E. Yaz

APPLICATIONS OF SINGULAR VALUE DECOMPOSITION TO THE APPROXIMATE PARTIAL MINIMAL REALIZATION PROBLEM....783
G. M. Pitstick, J. R. Cruz and R. J. Mulholland

IV-E: ROBUSTNESS AND SENSITIVITY THEORY

Chairman: P. R. Khargonekar
University of Minnesota

A NEW METHOD FOR ROBUST STABILITY....785
Q. Huang and R. Liu

REMARKS ON WEIGHTED SENSITIVITY MINIMIZATION....787
A. Pascoal and P. P. Khargonekar

ON THE PROBLEM OF SIMULTANEOUSLY STABILIZING A ONE PARAMETER FAMILY OF SINGLE INPUT SINGLE OUTPUT LINEAR DYNAMICAL SYSTEMS ... 795
B. K. Ghosh

INNER-OUTER FACTORIZATIONS ARISING IN H^∞ OPTIMIZATION PROBLEMS ... 797
B.-C. Chang

NECESSARY AND SUFFICIENT CONDITION FOR STABILITY OF PERTURBED SYSTEM ... 806
Q. Huang and R. Liu

ROBUST COMPENSATOR DESIGN BASED ON INPUT-OUTPUT INFORMATION ... 816
D. Cobb

MATCHING CONDITIONS FOR OUTPUT FEEDBACK CONTROL OF UNCERTAIN LINEAR SYSTEMS ... 819
B. R. Barmish

V-A: DATA STRUCTURES

Chairman: D. T. Lee
Northwestern University

PARALLEL INTERPOLATION SEARCH ... 821
J. H. Reif and D. E. Willard

SELF-ORGANIZATION AND ANNEALING ... 830
D. Richards

AN IMPLEMENTATION OF ORDERED PRIORITY QUEUES ... 840
S.-H. S. Huang

RANDOMIZED SEARCH IN ORTHOGONAL LISTS AND ON PERIPHERAL STORAGE DEVICES ... 850
W. H. Janko

V-B: COMMUNICATIONS

Chairman: E. Geraniotos
University of Maryland

FREE DISTANCES OF CONTINUOUS PHASE MODULATIONS UNDER EXTREME FILTERING ... 859
N. Seshadri and J. B. Anderson

PERFORMANCE OF DIVERSITY/REED-SOLOMON CODED SSFH COMMUNICATIONS WITH ERRORS/ERASURES VIA RATIO THRESHOLDING ... 869
K. G. Castor and W. E. Stark

DELAYED DECISION FEEDBACK SEQUENCE ESTIMATION 878
A. Duel and C. Heegard

A SIMPLE BIVARIATE DENSITY REPRESENTATION 888
P. K. Willett and J. B. Thomas

V-C: FOUNDATIONS OF COMPUTING

Chairman: M. C. Loui
University of Illinois

INCLUSION RELATIONSHIPS AMONG PERMUTATION PROBLEMS 898
D. P. Bovet and A. Panconesi

NONDETERMINISTIC REDUCIBILITIES AND SPARSE COMPLETE SETS 906
H.-L. Sung

A THEORY OF CLOSED COVERS: TO DECIDE PROGRESS OF COMMUNICATING FINITE STATE MACHINES 914
M. G. Gouda, L. E. Rosier and H.-C. Yen

AN EVENT ALGEBRA FOR THE STUDY OF LIVENESS IN SELF-TIMED COMPUTATIONAL NETWORKS 923
R. Melhem

V-D: DISCRETE EVENT SYSTEMS

Chairman: B. Krogh
Carnegie-Mellon University

ON SUPERVISOR REDUCTION IN DISCRETE EVENT SYSTEMS 933
A. F. Vaz and W. M. Wonham

MODIFIED PETRI NETS FOR HIERARCHICAL SIMULATION AND CONTROL OF MANUFACTURING SYSTEMS 940
B. H. Krogh and C. L. Beck

ON THE COMPUTATION OF SUPREMAL CONTROLLABLE SUBLANGUAGES 942
F. Lin and W. M. Wonham

PASSIVE IDENTIFICATION OF GRAMMATICALLY CONSTRAINED SWITCHED SYSTEMS 951
M. McIntyre and M. Vidyasagar

ON THE USE OF TEMPORAL LOGIC IN CONTROL THEORY 961
J. G. Thistle and W. M. Wonham

V-E: COMPUTER SYSTEMS

Chairman: J. Abraham
University of Illinois

A LARGE COMMERCIAL MULTI MICROPROCESSOR SYSTEM 971
K. Ueda and J. Sugano

LOW COST BRANCH PREDICTION 979
D. A. Poplawski

AN EFFICIENT ARCHITECTURE FOR IMPLEMENTING SPARSE ARRAY VARIABLES 986
J. T. O'Donnell

INDEX OF AUTHORS 996

The Plenary Lecture entitled

THE COMPLEXITY OF PARALLEL COMPUTATION

was delivered by

Prof. Richard M. Karp
University of California at Berkley

First, definitions of the complexity of parallel computation were discussed. This included a description of certain models for parallel random access memories. Problems of class NC were considered. These are problems that can be solved very quickly (specifically, in an amount of time that grows as a polynomial in the *log* of the problem size), using a number of processors that grows as a polynomial in the problem size. Karp gave sketches of recently discovered algorithms that demonstrate that the following problems are in NC: evaluation of expressions (a randomized algorithm of Miller and Reif was also described that runs in O(log n) time using only O(n/log(n)) processors), finding maximal independent sets in a graph, and inversion of a matrix. Evidence was given that suggests that certain problems with polynomial-time complexity, such as the problem of finding the lexicographically first maximal independent set in a graph, are not in NC. Finally, recent results about using bounded degree networks to simulate more idealistic shared-memory models for parallel computation were discussed. A recurring theme in the talk was that randomization appears to play a key role in many of the fast algorithms for parallel computation.

B.H.

EXACT NONLINEAR FILTERS, EXPONENTIAL DENSITIES AND SEPARATION OF VARIABLES

FREDERICK E. DAUM
Raytheon Company
Boston Post Road
Wayland, MA. 01778

ABSTRACT

Conditions for the existence of exact finite dimensional filters for nonlinear estimation problems are derived using two main ideas: (1) separation of variables in partial differential equations, and (2) the generalized Darmois-Koopman-Pitman theorem for densities of exponential type. Explicit equations for the exact filter are also derived as a by-product. The measurements are made at discrete points in time, but the random process to be estimated evolves continuously in time. The relationship of these results to Brockett's theory of estimation Lie algebras is discussed.

I. INTRODUCTION

Kalman solved the linear estimation problem with Gaussian noise in 1960. Despite twenty-five years of intensive research, very few recursive filters for nonlinear or non-Gaussian problems have been discovered. Some examples of such filters are given in [10], [11], [13] and [18]. Considering the apparent difficulty of finding recursive filters, Brockett decided that it would be useful to develop a theory that could predict when a given estimation problem has such a solution [1]. This theory has been developed for continuous time observations using Lie algebraic techniques in [14]-[17]. In this paper, an analogous theory is derived for discrete time measurements without using Lie algebraic methods. In addition, for those cases in which a recursive filter exists, the explicit filter equations are provided as a by-product. In contrast, Lie algebraic methods have not yet produced such an explicit construction of recursive filters.

Consider the problem of estimating a $\mathbb{R}^n$-valued random variable, $x(t)$, that evolves in continuous time according to the Itô equation:

$$dx(t) = f(x(t), t)dt + G(t)dw \tag{1}$$

given $\mathbb{R}^m$-valued discrete time measurements:

$$z_k = g(x(t_k), t_k, v_k) \tag{2}$$

in which v_k is an $\mathbb{R}^m$-valued random variable independent of x_k and with statistically independent values at distinct points in time. The probability density of z_k conditioned on $x_k = x(t_k)$ is assumed to be given, and it is denoted by $p(z_k|x_k)$ for $k = 1, 2, \ldots, N$. For example, if $z_k = h(x(t_k), t_k) + v_k$, and v_k is a Gaussian zero-mean random variable:

$$p(z_k|x_k) = c_1 \exp[-\tfrac{1}{2}(z_k-h(x_k, t_k))^T R_k^{-1}(z_k-h(x_k, t_k))]$$

where c_1 is a positive real number, and $R_k = E(v_k v_k^T)$ is assumed to be a given positive definite mxm matrix. In this paper, conditions are derived for the exact optimal filter to have a fixed finite dimension. This derivation is based on two main ideas: (1) the classical notion of separation of variables in partial differential equations (PDEs), and (2) the generalized Darmois-Koopman-Pitman theorem for densities of exponential type (see [4]-[8]). These conditions take the form of a set of PDEs. In a given coordinate system, Brockett's Lie algebraic criterion for the existence of finite dimensional filters [1] can also be expressed as a system of PDEs. Moreover, Miller [3] has developed a Lie algebraic theory for the separation of variables in second order linear PDEs. Figure 1 illustrates the relationship between these ideas. In this paper, no Lie algebraic theory is used, but rather the results are obtained by direct manipulation of the Fokker-Planck PDE. This should make the results accessible to a wider audience, considering that Lie algebras are not yet the stock in trade of most engineers.

In this paper, the random process, $x(t)$, evolves in continuous time, but the measurements are made in discrete time. This "hybrid" set-up contrasts with most filtering theory, which deals with continuous time random processes and continuous time measurements, or else discrete time random processes and discrete time measurements. A notable exception to the norm is the book by Jazwinski [19]. The reasons for this hybrid approach are three-fold. First, most physical processes are conveniently modelled in terms of differential equations (object motion, electrical circuits, chemical processes, nuclear reactions, biological phenomena, etc.). Second, the continuous time model of $x(t)$ allows the use of the elegant formalism of the Fokker-Planck equation. Third, in practical applications, filtering algorithms are implemented with digital computers, owing to their superior flexibility, accuracy and dynamic range as compared to currently available analog devices. The use of a digital computer requires that the measurements are made in discrete time. These points are reinforced by the recent survey of Kalman filter applications [20, p. 501].

II. MAIN RESULTS

Consider the nonlinear estimation problem stated in the Introduction. Suppose that $x(t)$ has a nowhere vanishing unconditional probability density, $p(x, t)$, that is sufficiently smooth to satisfy the Fokker-Planck equation. Furthermore, assume that the density $p(z_k|x_k)$ is also smooth and nowhere vanishing. Finally, suppose that there exist $M \times M$ matrices $A = A(t)$ and $B_j = B_j(t)$ $(j = 1, \ldots, M)$, such that the following set of PDEs has a solution for $u = u(x, t_k)$ and $\theta = \theta(x, t)$ and $\|\theta(x, t)\|$ is nowhere vanishing:

$$\frac{\partial\theta}{\partial t} = \frac{\partial\theta}{\partial x}(Qr^T - f) + \frac{1}{2}\xi - A\theta \tag{3}$$

$$\frac{1}{2}\frac{\partial\theta}{\partial x} Q\left(\frac{\partial\theta}{\partial x}\right)^T = \sum_{j=1}^{M} \theta_j B_j \tag{4}$$

$$\frac{\partial\eta}{\partial x} u + \eta\frac{\partial u}{\partial x} = -\frac{\partial\theta^\#}{\partial x} q_k - \theta^\# \frac{\partial q_k}{\partial x} \tag{5}$$

in which $r = r(x, t) = \dfrac{\partial p(x, t)}{\partial x}/p(x, t)$, $\xi_j = \mathrm{Tr}\left(Q \dfrac{\partial^2\theta_j}{\partial x^2}\right)$,

$\xi = (\xi_1, \ldots, \xi_M)^T$, $\theta = (\theta_1(x, t), \ldots, \theta_M(x, t))^T$,

$u = (u_1(x), \ldots, u_M(x))^T$, and $Q = G\,G^T$.

Moreover, $q_k = q(x, z_k) = \log_e p(z_k|x) - \log_e p(z_k|x_0)$, in which x_0 is a fixed but arbitrary point in $\mathbb{R}^n$. Also, $\eta = I - \theta\theta^T/\theta^T\theta$ and $\theta^\# = \theta/\theta^T\theta$.

Theorem: Under the conditions stated above, the exact unnormalized probability density of $x(t)$ conditioned on the sigma algebra generated by the set of measurements up to and including time t_k, $Z_k = \{z_1, z_2, \ldots, z_k\}$, is given by:

$$p(x, t|Z_k) = p(x, t) \exp [\theta^T(x, t) \psi(t)] \tag{6}$$

in which $\psi(t)$ is an $\mathbb{R}^M$-valued function of Z_k and time that evolves according to the following ordinary differential equation between measurements ($t_{k-1} \leq t \leq t_k$):

$$\frac{d\psi(t)}{dt} = A^T(t)\psi(t) + \Gamma(t) \tag{7}$$

in which $\Gamma = (\Gamma_1, \ldots, \Gamma_M)^T$ where $\Gamma_j = \mathrm{Tr}(\psi\psi^T B_j)$ for $j = 1, \ldots, M$. Moreover, the value of $\psi(t_k)$ after a measurement is related to the value immediately before a measurement (denoted by $\overline{\psi}(t_k)$) as follows:

$$\psi(t_k) = \overline{\psi}(t_k) + \eta(x_o, t_k)\, u(x_o, t_k) \tag{8}$$

The initial condition for (7) at time t_k is given by (8), and the value of $\overline{\psi}(t_k)$ in (8) is the solution of (7) over the interval $t_{k-1} \leq t \leq t_k$. If there were no previous measurements then the initial condition for (8) is $\overline{\psi}(t_1) = 0$. This filter is M-dimensional, and the sufficient statistic is the $\mathbb{R}^M$-valued quantity $\psi(t)$. Given $\theta(x, t_k)$, equation (5) is a first order quasi-linear PDE in $u(x, t_k)$, and therefore $u(x, t_k)$ can be obtained by solving an ODE via the method of characteristics.

Remark 1: The practical significance of these equations is that the estimation problem is factored into two parts: (1) the off-line calculation of $\theta(x, t)$ which does not depend on the measurements Z_k, and (2) the on-line calculation of $\psi(t)$ via (7) and (8), which depends on the measurements. The equations for $\theta(x, t)$ are PDE's which can be solved by numerical methods if required, whereas $\psi(t)$ is governed by an ODE, which could also be solved by numerical methods if necessary. The basic idea is that engineers might be willing to solve ODEs on-line in real-time (as is commonplace nowadays with extended Kalman filters), but they are generally unable to solve PDEs on-line in real-time.

Remark 2: The conditions given above are sufficient for the estimation problem to have a finite dimensional solution. Note that these results are formal, in the sense that there is no a priori guarantee that (7) has a solution. Intuitively something like controllability and observability must be assumed to guarantee existence of a solution to (7), similar to the results given in [10]. Furthermore, the existence of a solution to equation (3)-(5) is a necessary condition for the estimation problem to have a finite dimensional sufficient statistic. Finally, the condition that $p(x, t)$ is non-vanishing on $\mathbb{R}^n$ can be relaxed substantially. In particular, a theory can be constructed under the assumption that the subset of $\mathbb{R}^n$ on which $p(x, t)$ is non-vanishing is a finite union of connected subsets of $\mathbb{R}^n$.

III. SKETCH OF DERIVATION

The generalized Darmois-Koopman-Pitman theorem essentially says that for smooth nowhere vanishing densities, a sufficient statistic of fixed finite dimension will exist if and only if the conditional density, $p(x_k|Z_k)$, is from an exponential family [4]-[8]. The smoothness requirement is not merely a technical detail, but rather it has important practical implications. For example, exact recursive filters with fixed finite dimensions can be constructed for problems with non-smooth densities (e.g., uniformly distributed measurement noise and process noise) for which the conditional density is not from an exponential family (e.g., see [13]). It is easy to show that there is no loss in generality by assuming the form (6). The crucial conditions that we want to guarantee are that: (1) ψ is not a function of x, and (2) $\theta(x, t)$ is not a function of the measurements. This corresponds to the classical idea of "separation of variables" in PDE theory. Next, consider the Fokker-Planck equation corresponding to (1):

$$\frac{\partial p}{\partial t} = - \frac{\partial p}{\partial x} f - p \, \mathrm{Tr}\left(\frac{\partial f}{\partial x}\right) + \frac{1}{2} \mathrm{Tr}\left(Q \frac{\partial^2 p}{\partial x^2}\right) \tag{9}$$

in which $p = p(x, t|Z_k)$. The Fokker-Planck equation (9) governs the evolution of the conditional density of x(t) between measurements ($t_k \leq t \leq t_{k+1}$). Substituting (6) into (9), followed by several pages of elementary calculations results in:

$$\left(\frac{\partial \theta}{\partial t}\right)^T \psi + \theta^T \frac{d\psi}{dt} = \psi^T \frac{\partial \theta}{\partial x} [Q\, r^T - f] + \frac{1}{2} \psi^T \xi + \frac{1}{2} \psi^T \frac{\partial \theta}{\partial x} Q \left(\frac{\partial \theta}{\partial x}\right)^T \psi \tag{10}$$

Direct manipulation of (10) shows that it "separates" into an ODE for $\psi(t)$ and a PDE for $\theta(x, t)$ under the conditions assumed earlier. In particular, inspection of (3)-(4) shows that $\theta(x, t)$ is manifestly independent of $\psi(t)$ and Z_k. Moreover, substituting (3)-(4) into (10) leads directly to (7), and hence $\psi(t)$ is independent of x and $\theta(x, t)$. The basic strategy for showing that (3)-(4) are necessary conditions is outlined as follows. First, differentiate (10) twice w.r.t. ψ. A standard "separation of variables" argument from PDE theory shows that this implies (4) and (7). Next, substituting (4) and (7) into (10), and differentiating w.r.t. ψ results in (3).

Bayes rule implies that the following equation is satisfied as the result of a measurement at time t_k:

$$\theta^T(x,t_k)\psi = \theta^T(x,t_k)\, \overline{\psi} + q(z_k,x,t_k) \tag{11}$$

The most general solution of this linear equation is given by:

$$\psi - \overline{\psi} = \theta^{\#} q + \eta\, u \tag{12}$$

in which $u = u(x,t_k)$ is any $\mathbb{R}^M$-valued function of x. The general solution of linear equations, such as (11), is given in [22, pp. 23-24]. The requirement that $\Delta\psi = \psi - \overline{\psi}$ is functionally independent of x is guaranteed by setting $\frac{\partial \Delta\psi}{\partial x} = 0$ for all x in $\mathbb{R}^n$. This necessary and sufficient condition corresponds to (5). A more detailed derivation will be published elsewhere.

IV. DISCUSSION OF RESULTS

It is easy to verify that the conditions derived here are sufficiently general to cover the new filters reported in [10]. Moreover, the theory in [10] represents a unified treatment of the Kalman filter, the Beneš filter with discrete time measurements [11], and other nonlinear estimation problems including those with the generalized Maxwell-Boltzmann distribution. In particular, all of the filters derived in [10] have conditional densities of exponential type, with dimension $M = n + n(n + 1)/2$ for the sufficient statistic. On the other hand, the sufficient conditions derived here do not require that $M = n + n(n + 1)/2$.

For any practical nonlinear estimation problem, it is extremely unlikely that one can find A and B_j ($j = 1, \ldots, M$) that satisfy (3)-(5) exactly. This is the same situation that obtains in the case of the Kalman filter. Estimation problems with exactly linear dynamics and linear measurements are virtually non-existent. The wide spread use of the "extended" Kalman filter derives from the ease with which a given nonlinear problem can be approximated by a linear problem. Fortunately, a similar approach can be used here. In particular, equations (3)-(5) could serve as the starting point for an approximation method. For example, a convenient parameterization of $\theta(x, t)$ could be selected, and a Ritz-Galerkin technique could be used to approximate a solution of (3)-(5) by varying A, B_j and the parameters defining $\theta(x, t)$. Further details of this novel approximation technique will be reported elsewhere.

There are two main differences between these results and the Lie algebraic theory developed in [14]-[17]. First, this paper provides explicit filtering equations (6)-(8), whereas there is no analogous result from the Lie algebraic theory. Second, the Lie algebraic theory is "coordinate free", whereas the results given here are not. These two facts are intimately related. In particular, explicit filter equations can be obtained here because a special coordinate system was chosen in which the Fokker-Planck equation "separates". One can think of $\theta(x, t)$ as defining a set of preferred coordinates. Intuitively, the purpose of solving the PDE's (3)-(4) is to find this preferred coordinate system.

A growing number of people believe that differential geometry has much to say about finite dimensional nonlinear filters. In particular, the ideas of Brockett [1] and Mitter [2] have been developed in [14]-[17]. On the other hand, it might appear that the methodology of this paper is not related to differential geometry at all. However, this is not the case. For example, the basic mathematical tool that was used to prove the Darmois-Koopman-Pitman theorem is the implicit function theorem, which is one of the fundamental results of differential geometry! Moreover, Lie theory and differential geometry have much to say about the conditions under which (3)-(5) will have exact solutions. First, necessary and sufficient conditions for the existence of a solution to a set of first order PDE's, such as (5), are given in [9] (Chapter 6, Vol. I). Second, the original motivation for much of Lie's work was to find a geometrical characterization of the simultaneous solution of a set of linear first order PDEs (see [12], Chapter V).

ACKNOWLEDGEMENT

I would like to thank Dr. J. Beard for several helpful discussions concerning pseudo-inverses.

REFERENCES

[1] R. W. Brockett, "Classification and Equivalence in Estimation Theory," Proc. 1979 IEEE Conf. on Decision and Control, pp. 172-175, Dec. 1979.

[2] S. K. Mitter, "On the Analogy Between Mathematical Problems of Non-Linear Filtering and Quantum Physics," pp. 163-216, Ricerche diAutomatica, Vol. 10, No. 2, Dec. 1979.

[3] W. Miller, Symmetry and Separation of Variables, Reading, Ma.: Addison-Wesley Publishing Co., 1977.

[4] E. W. Barankin, and A. P. Maitra, "Generalization of the Fisher-Darmois-Koopman-Pitman Theorem on Sufficient Statistics," Sankhyā, Ser. A, Vol. 25, pp. 217-244, Sept. 1963.

[5] G. Sawitzki, "Finite Dimensional Filter Systems in Discrete Time," Stochastics, Vol. 5, pp. 107-114, 1981.

[6] G. Sawitzki, "Exact Filtering in Exponential Families: Discrete Time", pp. 554-558, Stochastic Control Theory and Stochastic Differential Systems, ed. by M. Kohlmann and W. Vogel, New York: Springer-Verlag, 1979.

[7] E. J. G. Pitman, "Sufficient Statistics and Intrinsic Accuracy", Proc. Camb. Phil. Soc., Vol. 32, pp. 567-579, 1936.

[8] B. O. Koopman, "On Distributions Admitting a Sufficient Statistic," Trans. Amer. Math. Soc., Vol. 39, pp. 399-409, 1936.

[9] M. Spivak, Differential Geometry, 2 Vols., 2nd edition, Berkeley: Publish or Perish, Inc., 1979.

[10] F. E. Daum, "Exact Finite Dimensional Nonlinear Filters," 24th IEEE Conf. on Decision and Control, Dec. 1985.

[11] F. E. Daum, "Exact Finite Dimensional Nonlinear Filters for Continuous Time Processes with Discrete Time Measurements," Proc. 1984 IEEE Conf. on Decision and Control, pp. 16-22, Dec. 1984.

[12] A. Cohen, An Introduction to the Lie Theory of One-Parameter Groups, New York: G. E. Stechert & Co., 1931.

[13] L. D. Servi and Y.-C. Ho, "Recursive Estimation in the Presence of Uniformly Distributed Measurement Noise," IEEE Trans. Automatic Control, Vol. AC-26, pp. 563-565, April 1981.

[14] M. Hazewinkel and S. I. Marcus, "On Lie algebras and finite dimensional filtering," Stochastics, Vol. 7, pp. 29-62, 1982.

[15] D. Ocone, "Nonlinear filtering problems with finite dimensional estimation algebras," Proc. Joint Automatic Control Conf., 1980.

[16] R. W. Brockett, "Nonlinear systems and nonlinear estimation theory," in Stochastic Systems: The Mathematics of Filtering and Identification and Applications, ed. M. Hazewinkel and J. C. Willems, Reidel, Dordrecht, The Netherlands, 1981.

[17] M. Chaleyat-Maurel and D. Michel, "Des Resultats de Non Existence de Filtre de Dimension Finie," Stochastics, Vol. 13, pp. 83-102, 1984.

[18] V. E. Beneš, "Exact Finite-Dimensional Filters for Certain Diffusions with Nonlinear Drift," Stochastics, Vol. 5, pp. 65-92, 1981.

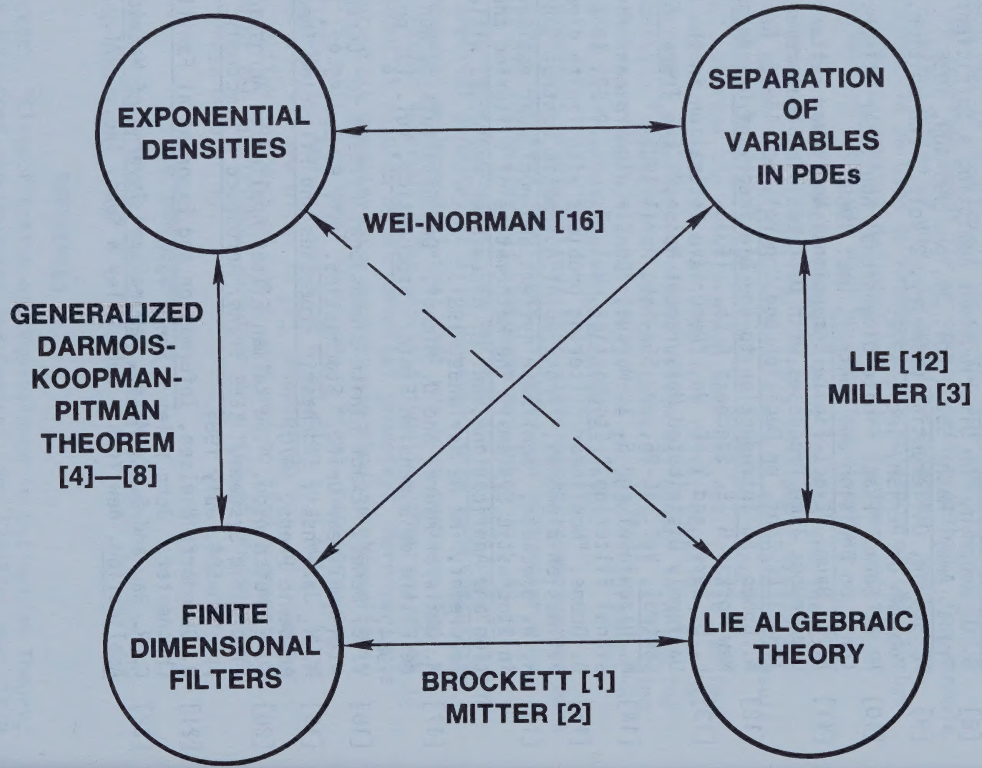
[19] A. H. Jazwinski, Stochastic Processes and Filtering Theory, New York: Academic Press, 1970.

[20] C. E. Hutchinson, "The Kalman Filter Applied to Aerospace and Electronic Systems," IEEE Trans. Aerospace & Electronic Systems, pp. 500-504, July 1984.

[21] O. Barndorff-Nielsen, Information and Exponential Families, Chichester: John Wiley & Sons, 1978.

[22] C. R. Rao and S. K. Mitra, Generalized Inverse of Matrices and Its Application, New York: John Wiley & Sons, Inc., 1971.

FIGURE 1 CIRCLE OF IDEAS RELATING TO FINITE DIMENSIONAL FILTERS


EXPONENTIAL DENSITIES
SEPARATION OF VARIABLES IN PDEs
WEI-NORMAN [16]
GENERALIZED DARMOIS-KOOPMAN-PITMAN THEOREM [4]—[8]
LIE [12]
MILLER [3]
FINITE DIMENSIONAL FILTERS
LIE ALGEBRAIC THEORY
BROCKETT [1]
MITTER [2]

FINITE DIMENSIONAL FILTERS FOR A CLASS OF NONLINEAR SYSTEMS THAT ARE IMMERSIBLE IN A LINEAR SYSTEM .

J. LEVINE
Centre d'Automatique et d'Informatique, Ecole des Mines de Paris
FONTAINEBLEAU, FRANCE.

ABSTRACT: We study the finite dimensionality of the filter associated to systems without dynamical noise. A necessary and sufficient condition for the existence of a finite dimensional filter is obtained , providing an explicit formula for the minimal filter. Furthermore, this condition is proved to be equivalent to the finite dimensionality of the Estimation algebra and to the immersibility in a linear system.

Introduction.

After R. Brockett [2], V. Benes [1], S. Mitter [14], D. Ocone [15], and others, (see also the beautiful Survey paper by S. Marcus [13]), most of the works on Finite Dimensional Filters (shortly FDF) make an explicit use of the finite dimensionality of the Estimation algebra (the formal Lie algebra generated by the drift and diffusion operators of the Duncan-Mortensen-Zakai (D-M-Z) equation).
A general method, outlined by Brockett [2], consists in finding sufficient conditions for finite dimensionality of the Estimation algebra and obtain a filter by the Wei-Norman integration technique (see also [17]). In fact, this program is generally untractable.

Another stream [3], [13], [14], consists in classifying the transformations that leave the Estimation algebra unchanged. Diffeomorphic changes of coordinates in the state-space and gauge transformations are among those invariant transformations.

A natural question raises then wether one can describe those invariant transformations directly from the system's input-output mapping rather than from the Estimation algebra. The purpose of this paper is precisely to give a complete answer to this question in the particular class of systems without dynamical noise.

Note that in applications, it may be interesting to modelize systems without dynamical noises in at least two situations: processes with short life duration, or short observations' histories; processes with very slow observations compared to the dynamics.

As a result, a necessary and sufficient condition for such a system, noted Σ , to have a FDF is that there exists an immersion from Σ to a linear system. Furthermore, it turns out that this condition is equivalent to the finite dimensionality of the Estimation algebra of Σ .

Another interesting consequence of the immersion's approach, is that the explicit filtering equations are obtained in a particularly straigtforward way, without using the Wei-Norman equations.

These results generalize the heuristic ones of [16] where the Estimation algebra is assumed nilpotent.

The results are stated without proofs. The detailed proofs are in [10].

1. The unnormalized conditional density

We consider the system Σ :

$$\left.\begin{aligned} dx_t &= f(x_t)dt, \quad x_0 : \text{random with density } \mu_0, \\ dy_t &= h(x_t)dt + dv_t, \quad y_0 = 0, \end{aligned}\right\} \ (\Sigma)$$

where we assume that $x_t \in \mathbb{R}^n$, $y_t \in \mathbb{R}^p$, x_0 : random with density μ_0 with respect to Lebesgue measure of $\mathbb{R}^n$, $\mu_0 \in C^\infty(\mathbb{R}^n ; \mathbb{R}_+)$, and v_t is a standard Brownian motion of $\mathbb{R}^p$, dv_t being its Stratonovitch differential.
We denote ' for transposition, $Y_t = \{y_s \,/\, 0 \leq s \leq t\}$, and we also assume that f is a C^∞ vector field on $\mathbb{R}^n$, with linear growth and bounded divergence, and that h is a C^∞ function from $\mathbb{R}^n$ to $\mathbb{R}^p$.
The assumptions on f ensure that the flow $(x,t) \to X_t(x)$ of the differential equation $\dot{x} = f(x)$, with $X_0(x) = x$, satisfies : X_t is a C^∞ diffeomorphism of $\mathbb{R}^n$ $\forall t \geq 0$.

Theorem 1: The unnormalized conditional density of x_t knowing Y_t , noted π_t^Y , exists, $\pi_t^Y \in C^\infty(\mathbb{R}^n ; \mathbb{R}_+)$,$\forall t,Y$, and:

$$\pi_t^Y(x) = \exp \sum_{i=1}^{p} \Big[\int_0^t h_i(X_{s-t}(x))dy_s^i - 1/2 \int_0^t h_i^2(X_{s-t}(x))ds \Big] \tag{1.1}$$

$$\left|\det\frac{\partial X_{-t}}{\partial x}(x)\right| \mu_0(X_{-t}(x)) \quad \forall t\geq 0, \ \forall x \in \mathbb{R}^n, \ Y \text{ a.s.}$$

Furthermore, π_t^Y solves the D-M-Z equation (in Stratonovitch form):

$$\left.\begin{aligned} d\pi_t^Y &= [-\operatorname{div}(f\pi_t^Y) - 1/2 \sum_{i=1}^{p} h_i^2\pi_t^Y]dt + \sum_{i=1}^{p} h_i\pi_t^Y dy_t^i & \text{(1.2.a)} \\ \pi_0^Y &= \mu_0 & \text{(1.2.b)} \end{aligned}\right\}$$

2. Existence of finite dimensional filters and their minimal realization

Definition 1: A FDF for Σ is a stochastic system on $\mathbb{R}^r$:

$$\left.\begin{aligned} d\xi_t &= a(\xi_t)dt + \sum_{i=1}^{p} b_i(\xi_t)dy_t^i & \text{(2.1.a)} \\ \pi_t^Y(x) &= \nu(t,x)\sigma(\xi_t,x) \quad \forall x \in \mathbb{R}^n \ , \quad Y \text{ a.s.} & \text{(2.1.b)} \end{aligned}\right\}$$

with $a,b_1,\dots,b_p \in C^\infty(\mathbb{R}^r;\mathbb{R}^r)$, a having linear growth, b_i bounded $\forall i$,

$\nu \in C^0(\mathbb{R}^n \times \mathbb{R}_+ ; \mathbb{R}_+)$,$\sigma \in C^\infty(\mathbb{R}^r \times \mathbb{R}^n ; \mathbb{R}_+)$,

and ξ_0 s.t. $\mu_0 = \nu(0,.)\sigma(\xi_0,.)$, with π_t^Y given by (1.1).

Our FDF's definition is, apart from integrability details, equivalent to the definition of universal filters of [4], (see also [13]), but appears more naturally with (1.1).

let us denote $L_f h_i(x)$ the Lie derivative of h_i along f :

$$L_f h_i(x) = \sum_{j=1}^{n} f_j(x) \frac{\partial h_i}{\partial x_j}(x) \quad , \quad i = 1,\dots,p, \tag{2.2}$$

$$L_f^k h_i = L_f (L_f^{k-1} h_i), \ \forall k \geq 1 \quad , \quad L_f^0 h_i = h_i ,$$

The following functional space H is often called the observations space of Σ (see [6]) and plays a crucial role :

$$H = \mathbb{R}\text{-Span}\{ L_f^k h_i \ / \ 1 \leq i \leq p, \forall k \geq 0 \} \tag{2.3}$$

Theorem 2: The three following assertions are equivalent:

(i) Σ has a FDF

(ii) $\dim H < \infty$

(iii) $\exists r \in \mathbb{N}$, $\exists \theta_1,\dots,\theta_r \in H$, $\exists A \in \mathbb{R}^{r\times r}$, $\exists B_1,\dots,B_p \in \mathbb{R}^{1\times r}$ such that:

$$L_f \theta = A\theta \tag{2.4.a}$$

$$h_i = B_i \theta \ , \quad \forall i = 1,\dots,p \tag{2.4.b}$$

with $\theta(x) = (\theta_1(x),\dots,\theta_r(x))'$, ' denoting transposition.

An easy consequence of this result is the following :

Theorem 3: If $\dim H = r < \infty$, the minimal realization of Σ 's filter is linear of dimension r, and explicitly given by:

$$d\xi_t = -A'\xi_t dt + \sum_{i=1}^{p} B_i' dy_t^i \ , \quad \xi_0 = 0 \tag{2.5.a}$$

$$\pi_t^Y(x) = \nu(t,x) \exp\Big(\sum_{i=1}^{r} \xi_t^i \theta_i(x) \Big) \tag{2.5.b}$$

with A, $B_1,\dots,B_p,\theta_1,\dots,\theta_r$ given in (iii) of theorem 2 and ν given by :

$$\nu(t,x) = \Big[\exp -1/2 \sum_{i=1}^{p} \int_0^t h_i^2(X_{s-t}(x)) ds\Big] \left| \det \frac{\partial X_{-t}}{\partial x}(x) \right| \mu_0(X_{-t}(x)). \tag{2.6}$$

Example 1: A polynomial sensor problem.

$n = 1,\ p = 1.$

$$\left.\begin{aligned} dx_t &= Fx_t dt && (2.7.a)\\ dy_t &= \sum_{i=0}^{k} \alpha_i x^i dt + dv_t && (2.7.b)\end{aligned}\right\}$$

the integer k being arbitrary. Assuming that $\alpha_0 \neq 0$, we have $\dim H = k + 1$ (if $\alpha_0 = 0$, then $\dim H = k$). A basis of H is given by: $\theta_i(x) = x^i$, $i = 0,..,k$, and one can easily check that:

$$L_f\theta_i = iFx^i\ ,\quad i = 0,\dots,k, \qquad h(x) = \sum_{i=0}^{k} \alpha_i x^i = \sum_{i=0}^{k} \alpha_i \theta_i(x).$$

Thus, the minimal realization of the filter of (2.7) has dimension k+1, and is given by:

$$d\xi_t^i = iF\xi_t^i dt + \alpha_i dy_t\ ,\quad \xi_0^i = 0\ ,\quad i = 0,\dots,k \qquad (2.8.a)$$

$$\pi_t^Y(x) = (\exp -1/2 \int_0^t (\sum_{i=0}^{k} \alpha_i x^i e^{-iF(t-s)})^2 ds) \qquad (2.8.b)$$

$$e^{-Ft}\ \mu_0(e^{-Ft}x)(\exp \sum_{i=0}^{k} \xi_t^i x^i)\ .$$

It is remarkable that $\forall k$, (2.7) has a FDF, whereas, with noisy dynamics, there is no FDF at least when $k = 3$ (cubic sensor problem [7]).

Example 2: A continuous-time version of the tracking problem of [12].

$n = 3$ and $p = 1$. The vector x is noted $(x^1, x^2, x^3)'$.

$$\left.\begin{aligned} \dot{x}_t &= \begin{pmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} x_t && (2.9.a)\\ dy_t &= -\frac{2x^1}{x^2} dt + dv_t && (2.9.b)\end{aligned}\right\}$$

It can be easily checked that H is generated by the 2 functions:

$$\theta_1(x) = \frac{x^1}{x^2} \quad , \quad \theta_2(x) = \frac{x^3}{x^2} .$$

The minimal filter is thus:

$$d\xi_t = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \xi_t + \begin{pmatrix} -2 \\ 0 \end{pmatrix} dy_t \quad , \quad \xi_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{2.10.a}$$

$$\pi_t^Y = (\exp - \frac{2t}{(x^2)^2}((x^1)^2 + tx^1x^3 + \frac{t^2}{3}(x^3)^2))\ \mu_0(x^1 - tx^3, x^2, x^3) \tag{2.10.b}$$

$$\exp(\xi_t^1 \frac{x^1}{x^2} + \xi_t^2 \frac{x^3}{x^2})$$

Note that the minimal realization of the filter is 2 dimensional, though the original system is 3 dimensional: this obviously follows from the unobservability of x^3 , and it results that x^1 and x^2 's estimates can be improved by filtering whereas the information about x^3 remains unchanged.

3. FDF, Immersion in a linear system and Estimation algebra.

Let us first recall what is meant by immersion in a linear system [5] , and by Estimation algebra [2]:

Definition 2: Σ is said to be immersible in the linear system:

$$dz_t = Az_t dt \quad , \quad z_0 \in \mathbb{R}^m \tag{3.1.a}$$

$$dy_t = Bz_t dt + dv_t \quad , \quad y_0 = 0 \tag{3.1.b}$$

if there exists a C^∞ mapping θ from $\mathbb{R}^n$ to $\mathbb{R}^m$ (m arbirary) satisfying:

$$h(X_t(x)) = Be^{At}\theta(x) \quad , \forall\, x \in \mathbb{R}^n \quad , \forall\, t \geqslant 0. \tag{3.2}$$

The mapping θ is called the immersion from Σ to (3.1.a), (3.1.b)

Definition 3: Let us denote $L_0, \ldots, L_p$ the following operators:

$L_0\varphi = L_f\varphi - 1/2 \| h \|^2\varphi \quad , \quad L_i\varphi = h_i\varphi \quad ,i = 1,\ldots,p, \quad \forall\, \varphi \in C^\infty(\mathbb{R}^n;\mathbb{R})$ (3.3)

The Estimation algebra of Σ is the formal Lie algebra generated by $L_0,\ldots,L_p$.

Theorem 4: The following conditions are equivalent:

(i) Σ has a FDF

(ii) Σ is immersible in a linear system

(iii) The Estimation algebra of Σ is finite dimensional

Furthermore, if there is an immersion from Σ to the linear system (3.1.a),(3.1.b), then one can find an isomorphism of Lie algebras from the Estimation algebra of Σ to the Estimation algebra of (3.1.a),(3.1.b).

Remark 1: In [16], the following special case of Theorem 4 was obtained: If the Estimation algebra is nilpotent then Σ has a FDF . Furthermore, the realization presented in this case is not generally minimal.

Remark 2: As a consequence of Theorems 3 and 4, it appears that there is a duality, in the classical meaning of the theory of linear systems , between the minimal filter (2.5.a) and the linear system (3.1.a),(3.1.b) in which Σ is immersed.

Remark 3: Theorem 4 proves, at least for the class of systems under consideration, that Brockett's conjecture is true as well as its converse, and on the other hand that the immersion is the largest class of changes of coordinates leaving the Estimation algebra invariant. Note that the result on the finite dimensionality of the Estimation algebra is probably not general (see [4]).

Remark 4 : In [10], other interpretations of the FDF property can be done in terms of Jakubczyk's realization theory for nonlinear systems ([8]), and in terms of the Malliavin calculus of variations (see for ex. [9]).

ACKNOWLEDGEMENTS : The author is indebted to Profs. E. Pardoux, M. Fliess and R. Marino for very helpful suggestions.

REFERENCES

[1] V.E. BENES :Exact finite dimensional filters for certain diffusions with nonlinear drifts. Stochastics 5, 1981, 65-92.

[2] R.W. BROCKETT :Remarks on finite dimensional nonlinear estimation. C. Lobry ed. "Analyse des systèmes". Astérisque 75, 76 (SMF), p. 47-56.

[3] R.W. BROCKETT :Classification and equivalence in estimation theory. 18 th CDC. Ft. Lauderdale FL, 19879, 172-175.

[4] M. CHALEYAT-MAUREL, D. MICHEL :Un théorème de non existence de filtre de dimension finie. CRAS, t. 296, p. 933-936, 1983.

[5] D. CLAUDE, M. FLIESS, A. IṠIDORI :Immersion, directe et par bouclage, d'un système non linéaire dans un linéaire. C.R.A.S. t. 296, Série I, 1983, 237-240.

[6] M. FLIESS, I. KUPKA :A finiteness criterion for nonlinear input-output differential systems. SIAM J. Cont. & Optim., 21, 5 (1983), 721-728.

[7] M. HAZEWINKEL, S.I. MARCUS, H.J. SUSSMANN :Non existence of exact finite dimensional filters for conditional statistics of the cubic sensor problem. Syst. Cont. letters, 3, 331-340, 1983.

[8] B. JAKUBCZYK :Existence and uniqueness of realizations of nonlinear systems. SIAM J. Cont. & Optim., 18, 4 (1980), 455-471.

[9] S. KUSUOKA, D.W. STROOCK :Applications of the Malliavin calculus. Part I. To appear.

[10] J. LEVINE :Finite dimensional filters for a class of nonlinear systems, immersion in a linear system, and stochastic realization. To appear.

[11] J. LEVINE, G. PIGNIE :Exact finite dimensional filters via systems realization for a class of discrete-time nonlinear systems. Systems & Control Letters 5 (1985) 403-412.

[12] J. LEVINE, G. PIGNIE :The finite dimensional filtering problem for a class of nonlinear discrete- time systems. Proc. of 9th IFAC Wqrld Congress. Budapest 1984.

[13] S.I. MARCUS :lgebraic and geometric methods in nonlinear filtering. SIAM J. Cont. & Optim. 22, 6 (1984) 817-844.

[14] S.K. MITTER :Geometric theory of nonlinear filtering. In Outils et modèles mathématiques en automatique, analyse des systèmes et traitement du signal, Part 3, I. Landau Ed. Editions du CNRS, Paris (1982), 37-60.

[15] D.L. OCONE :Topics in nonlinear filtering theory. Ph.D. Thesis, M.I.T., Cambridge, 1980.

[16] Z.S. ROTH, K.A. LOPARO :Optimal filter realization for a class of nonlinear systems with finite dimensional estimation algebra. Syst. Cont. letters, 4, 1, 1984, p. 23-26.

[17] W. WONG :New classes of finite dimensional filters. Systems & Control Letters, 3 (1983) 155-164.

A MINIMAX FILTERING THEORY FOR COLORED NOISE

RANDALL K. BAHR and JAMES A. BUCKLEW
Department of Electrical and Computer Engineering
University of Wisconsin, Madison, WI 53706

ABSTRACT

This paper considers estimation of non-random data corrupted by random noise. It is desired to find a linear estimator whose performance is best over a class of non-random signals. This estimator will minimize the maximum mean square error over a set of input signals whose energy at the output of a given linear filter lies below a pre-assigned value. Asymptotic expressions for the minimax estimator error are developed for the case of additive colored noise possessing a continuous power spectral density. An analogy between this non-random scheme and the Wiener filter is suggested.

I. INTRODUCTION

The techniques of Wiener filtering are fairly well understood in applications involving estimating and filtering of random data. However, the researcher who would rather consider the data as deterministic, but unknown, is confronted by a multitude of various techniques, with varying degrees of theoretical justification. Such analysis problems involving non-random signals arise in the biomedical realm in the processing of electro-encephalographic data, in sonar and geophysics with the analysis of acoustical signals scattered through dispersive media and in radar with its cyclostationary signal structures, to name just a few. In this paper, the estimation of a discrete time deterministic signal in arbitrary colored noise via techniques involving the Discrete Fourier Transform (DFT) is investigated. The estimator will be minimax over all signals whose power at the output of a linear filter is limited to a pre-assigned level. That is, a linear estimator will be developed which minimizes the maximum mean square error over a prescribed set of signals. This "linear filter" constraint will allow more freedom than is currently available with spline polynomial fits. Furthermore the linear estimator solution will be shown to have error asymptotically equal to a certain Wiener filter estimator, for a specific choice of random signal spectrum.

The problem of filtering a non-random signal in noise is closely related to that of smoothing noisy data with spline functions. This subject is extensively treated in the statistics literature, for example see [1], [2] and [3]. In particular, two papers which provided the impetus for our work are [4] and [5].

Non-random data techniques are useful in biomedical signal analysis, for example, smoothing of evoked potential responses. Averaging methods applicable to this type of signal are discussed in [6]. Modifications of Wiener techniques are developed in [7], [8], and further theoretical treatment is provided by [9].

In our paper, we approach the deterministic data in noise problem from a minimax point of view. In the next section the problem is precisely defined.

II. DEVELOPMENT

Consider the following problem: Observe n samples of a corrupted non-random sequence and construct the minimax estimator of the original sequence under some reasonable constraint. Specifically, let $\{x(k)\}_{k=0}^{n-1}$ be a signal corrupted by zero mean noise $\{\varepsilon(k)\}_{k=0}^{n-1}$.

$$y(k) = x(k) + \varepsilon(k) \qquad k = 0, 1, \ldots, n-1 \tag{2.1}$$

Denote $\vec{x} \triangleq (x(0), x(1), \ldots, x(n-1))' \varepsilon R^n$, $\vec{\varepsilon} \triangleq (\varepsilon(0), \varepsilon(1), \ldots, \varepsilon(n-1))' \varepsilon R^n$ and suppose $\mathbf{F} \triangleq \{F\colon \mathbf{R}^n \to \mathbf{R}^n,\ F$ is a linear estimator of $\vec{x}\}$. Define an error criterion

$$E(\vec{x}, \hat{\vec{x}}) = E\{\frac{1}{n} \sum_{k=0}^{n-1} (x(k) - \hat{x}(k))^2\} = E\{\frac{1}{n} |\vec{x} - F\vec{y}|^2\} \tag{2.2}$$

where $E\{\cdot\}$ denotes expectation, $\hat{\vec{x}}$ is any linear estimate of $\vec{x}$ based on data $\vec{y}$, and $|\cdot|$ is the usual Euclidean distance norm on $\mathbf{R}^n$. The linear minimax estimation problem is to find $F \varepsilon \mathbf{F}$ minimizing the largest error with $\|\vec{x}\|_s \leq \sqrt{P}$, $P > 0$ specified and $\|\cdot\|_s$, a semi-norm, to be given. The minimax error becomes

$$\min_{F \varepsilon \mathbf{F}} \ \max_{\|\vec{x}\|_s \leq \sqrt{p}} \ E(\vec{x}, F(\vec{y})) \ . \tag{2.3}$$

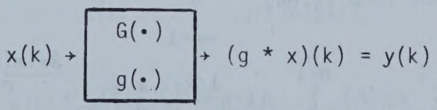
The signal $\vec{x}$ is constrained to have fixed energy P at the output of a linear filter. Let G be a complex valued function defined on [0,1]. Suppose M_1 is a positive constant such that $|G(x)| \leq M_1$, and further suppose $G^*(x) = G(1-x)$ for all $x \varepsilon [0,1]$. Thus G is seen to be the z-transform of some time series evaluated on the unit circle in the complex plane. Define

$$g(\tfrac{k}{n}) \triangleq \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} G(\tfrac{m}{n})\, e^{\frac{-2\pi imk}{n}} \tag{2.4a}$$

and thus

$$G(\tfrac{k}{n}) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} g(\tfrac{m}{n})\, e^{\frac{2\pi imk}{n}} \tag{2.4b}$$

$\{g(\frac{k}{n})\}_{k=0}^{n-1}$ is the impulse response of the network specified by G.



where * denotes circular convolution. Then

$$\sum_{k=0}^{n-1} y(k)^2 = \sum_{k=0}^{n-1} |G(\tfrac{k}{n})|^2 |X(k)|^2 \triangleq \|\vec{x}\|_S^2 \tag{2.5}$$

where $\{X(k)\}$ is the DFT of $\{x(k)\}$.

For the case that the corrupting noise $\{\varepsilon(k)\}$ is i.i.d. an exact expression for the filter structure and resulting minimax error may be stated.

Theorem 1: Under the observation model (2.1), for the case of additive white noise, the filter which meets the error criterion (2.3) under deterministic signal constraint (2.5) is given by

$$\hat{x}(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} (1 - \gamma^{1/2}|G(\tfrac{m}{n})|)_+ \, Y(m) \, e^{-\frac{2\pi ikm}{n}} \tag{2.6a}$$

and

$$Y(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} y(m) \, e^{\frac{2\pi imk}{n}} \tag{2.6b}$$

The minimum error is obtained by solving

$$\text{minimum error} = \min_{\gamma>0} \frac{1}{n} \{\gamma P + \sigma^2 \sum_{k=0}^{n-1} (1 - \gamma^{1/2}|G(\tfrac{k}{n})|)_+^2\} \tag{2.7}$$

where x_+ denotes the maximum of x and 0.

If $\{\varepsilon(k)\}$ is a colored noise sequence, i.e. consisting of n samples from a stationary random process with continuous spectral density $S(x)$, $S(x)$ not identically a constant, a closed form expression analogous to the above is not readily obtained. However, for the case of large sample size, an asymptotic error expression is available.

Theorem 2: The filter

$$\hat{x}(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} (1 - \gamma_n^{1/2}|G(\tfrac{m}{n})|)_+ Y(m) \, e^{-\frac{2\pi ikm}{n}} \tag{2.8a}$$

$$Y(k) = \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} y(m) \, e^{\frac{2\pi ikm}{n}} \tag{2.8b}$$

or substituting (2.8b) into (2.8a)

$$\hat{x}(k) = \sum_{\ell=0}^{n-1} y(\ell) \sum_{m=0}^{n-1} (1-\gamma^{1/2}|G(\tfrac{m}{n})|)_+ \, e^{\frac{2\pi im(\ell-k)}{n}} \, \frac{1}{n} \tag{2.8c}$$

where γ_n is chosen to minimize

$$\frac{1}{n}\{\gamma_n Pn + \sum_{k=0}^{n-1} S(\tfrac{k}{n})(1-\gamma_n^{1/2}|G(\tfrac{k}{n})|)_+^2\}$$

$$= \gamma_n P + \frac{1}{n}\sum_{k=0}^{n-1} S(\tfrac{k}{n})(1-\gamma_n^{1/2}|G(\tfrac{k}{n})|)_+^2 \qquad (2.9)$$

has minimax error converging to the asymptotic value

$$\min_{\gamma}\{\gamma P + \int_0^1 S(x)\,(1-\gamma^{1/2}|G(x)|)_+^2 dx\} \; . \qquad (2.10)$$

III. WIENER FILTER ANALOGY

Though the minimax estimation procedure derived here is applicable only to non-random signal sets, because the filter is designed to be best over an entire class of signals this suggests that in some sense the signal set could be taken as a random ensemble. This invites a comparison to the Wiener filter. The idea is to find a stationary random process, $\{x(k)\}_{k=-\infty}^{\infty}$, emulating the unknown deterministic signal. The random process will be constrained such that the power at the output of a discrete linear filter with impulse response $g(k) = \int_0^1 G(\eta)e^{-2\pi ik\eta}d\eta$ is limited to a pre-assigned value $P > 0$. This constraint is analogous to (2.5). From the infinite collection of power spectral densities of $\{x(k)\}$ which satisfy this requirement, the spectrum maximizing the minimum mean square point error will be chosen. The following theorem is obtained.

Theorem 3: The non-realizable discrete time Wiener filter which maximizes the minimum mean square point error for fixed noise spectrum $S_{\varepsilon\varepsilon}(\eta)$ is given by

$$H_0(\eta) = \frac{S_0(\eta)}{S_0(\eta)+S_{\varepsilon\varepsilon}(\eta)} = (1-\lambda^{1/2}|G(\eta)|)_+ \qquad (3.1)$$

where

$$S_0(\eta) = S_{\varepsilon\varepsilon}(\eta)(\frac{1}{\lambda^{1/2}|G(\eta)|} - 1)_+ \qquad (3.2)$$

is the worst case signal spectrum. Specifically,

$$\hat{x}(k) = \sum_{\ell=-\infty}^{\infty} y(\ell)h(k-\ell) \qquad (3.3a)$$

$$h(k) = \int_0^1 (1-\lambda^{1/2}|G(\eta)|)_+ \, e^{-2\pi i\eta k} \; . \qquad (3.3b)$$

Further, the maximum mean square error

$$\text{maximum mse} = \min_{\lambda>0} \{\lambda P + \int_0^1 S_{\varepsilon\varepsilon}(\eta)(1-\lambda^{1/2}|G(\eta)|)_+^2 \, d\eta\} \ . \qquad (3.4)$$

is identical to the asymptotic minimax error (2.10) for the deterministic data in noise problem.

IV. EXAMPLE

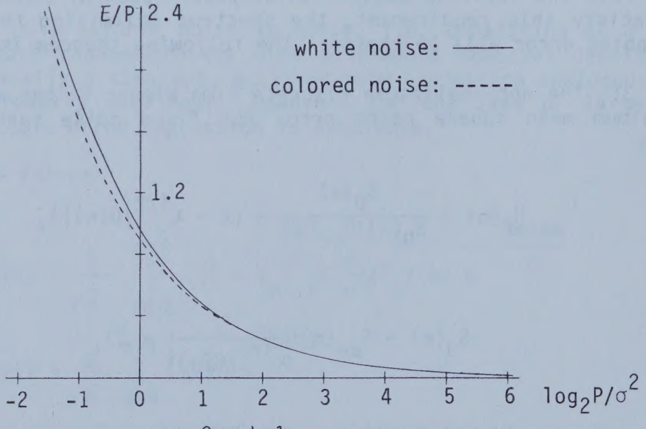
Consider signal sets whose total energy in a certain passband is limited. Let the linear network be described by

$$G(x) = \begin{cases} 1 & \frac{1}{8} \le x \le \frac{1}{4} \ , \ \ \frac{3}{4} \le x \le \frac{7}{8} \\ 0 & \text{otherwise} \end{cases}$$

and restrain the energy at the output by P. Graph 1 shows *E*/P for additive white noise of density σ^2 and also for additive colored noise with autocorrelation

$$R(k) = \begin{cases} \sigma^2 & k = 0 \\ \sigma^2/2 & k = \pm 1 \\ 0 & \text{otherwise} \end{cases}$$

and, consequently, $S(x) = \sigma^2(1 + \cos 2\pi x)$.



Graph 1

V. DISCUSSION AND CONCLUDING REMARKS

In this paper we have examined one technique of estimating a non-random signal in noise. It is assumed that though the signal is deterministic, the estimating procedure must handle any signal from a general class. This class consists of all signals whose energy at the output of a linear network is limited to a pre-specified value. The philosophy adopted here is to design a filter which minimizes the maximum possible error over this constraint class. The limiting value for large sample sizes of this minimax error was derived for additive colored noise, whose only restriction is that its spectral density exists and is continuous.

Comparison of the minimax filter (2.8c) and the Wiener filter (3.3) indicate their similar natures. For large n, the convolutional filter coefficients for the minimax filter (2.8c) "look like" the analogous coefficients of (3.3b) in the sense that the (Riemann) sums of (2.8c) approximate the integral of (3.3b). It is tempting to conclude that use of the Wiener filter on non-random data will yield the asymptotic minimax error of (2.10). Unfortunately, the authors have not been able to establish this or exhibit a counterexample.

APPENDIX I.

Proof of Theorem 1:

Consider a slightly modified DFT defined on $\mathbf{R}^n$:

$$X(k) \triangleq \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} x(m)\, e^{\frac{2\pi imk}{n}} \tag{A1.1a}$$

$$x(k) \triangleq \frac{1}{\sqrt{n}} \sum_{m=0}^{n-1} X(m)\, e^{-\frac{2\pi imk}{n}} \tag{A1.1b}$$

where $\vec{X} \triangleq (X(0), X(1), \ldots, X(n-1))' \varepsilon\ \mathbf{C}^n$, is the DFT of $\vec{x}\ \varepsilon\ \mathbf{R}^n$. This admits the following matrix representation

$$U \triangleq \left(\left(\frac{e^{-\frac{2\pi ikm}{n}}}{\sqrt{n}}\right)\right) \tag{A1.2}$$

then

$$\vec{X} = U^*\vec{x} \quad \text{and} \quad \vec{x} = U\vec{X} \tag{A1.3}$$

where * denotes transposition and conjugation. It is well known that U is orthogonal, i.e. $U^*U = I$. Define

$$\|\vec{X}\|_s \triangleq \left[\sum_{m=0}^{n-1} a_m |X(m)|^2\right]^{1/2} \tag{A1.4}$$

where $a_m = |G(\frac{m}{n})|^2$.

To derive the minimax estimator express the error in the frequency domain. Realizing **F** is simply the class of all $n \times n$ real matrices, denote F as the generic member and reformulate $E(\vec{x}, \hat{\vec{x}})$ in terms of the exponential basis.

$$\begin{aligned} E(\vec{x},\hat{\vec{x}}) &= E\,\frac{1}{n}\|\vec{x} - F\vec{y}\|^2 \\ &= \frac{1}{n}\left[|(I-F)\vec{x}|^2 + \mathrm{Tr}F'FR\right] \\ &= \frac{1}{n}\left[|(I-B)\vec{X}|^2 + \mathrm{Tr}(B^*BU^*RU)\right] \end{aligned} \tag{A1.5}$$

where I is the n × n identity matrix, $R \triangleq$ covariance $(\vec{\varepsilon})$ and $B \triangleq U^{*}FU$. Define Λ_a such that

$$\|\vec{x}\|_s \le \sqrt{P} \iff \sum_{k=0}^{n-1} a_k |X(k)|^2 \le P \iff \vec{X}^{*}\Lambda_a X \le P \tag{A1.6}$$

where $\Lambda_a \triangleq \mathrm{diag}(a_0, a_1, \ldots, a_{n-1})$.

Now reformulate the minimax problem. Find B, an n × n complex matrix, to minimize the maximum mean square error under the constraint $\|\vec{x}\|_s \le \sqrt{P}$. This minimax error becomes

$$\min_{B} \max_{\vec{X}^{*}\Lambda_a \vec{X} \le P} \frac{1}{n}\{|(I-B)\vec{X}|^2 + \mathrm{Tr}(B^{*}BU^{*}RU)\} \tag{A1.7}$$

where $\vec{X}$ is the DFT of $\vec{x}$, R is the covariance matrix of the additive colored noise $\vec{\varepsilon}$, and U is the exponential basis matrix. The solution to this problem requires a slight extension of a lemma given by Speckman [4], under the assumption $R = \sigma^2 I$.

Lemma 1: Let $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and $S = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2)$ be non-null diagonal matrices with $\lambda_k \ge 0$, $\sigma_k^2 \ge 0$, $k = 1, 2, \ldots, n$. Let $\beta \in \mathbf{C}^n$. The solution to the minimax problem

$$\min_{B} \max_{\beta^{*}\Lambda\beta \le \alpha^2} \{|(I-B)\beta|^2 + \mathrm{tr}(B^{*}BS)\}$$

where the minimum which is taken over n × n complex matrices B is achieved when $B = B(\gamma) = \mathrm{diag}((1 - \sqrt{\gamma\lambda_k}\,)_+)$ for some constant $\gamma > 0$. Where x_+ denotes the maximum of x and 0.

Consequently, designate $D = \mathrm{diag}(d_1, \ldots, d_n)$ with $d_k \triangleq (1 - \sqrt{\gamma a_k})_+$ and the minimax solution becomes

$$= \frac{1}{n}\{|(I-D)\vec{X}|^2 + \sigma^2\,\mathrm{Tr}(D^{*}D)\}$$

$$= \frac{1}{n}\{\gamma P + \sigma^2 \sum_{k=0}^{n-1} (1 - \sqrt{\gamma a_k}\,)_+^2\} \tag{A1.8}$$

for some $\gamma > 0$ or

$$\text{minimum error} = \min_{\gamma > 0} \frac{1}{n}\{\gamma P + \sigma^2 \sum_{k=0}^{n-1} (1 - \sqrt{\gamma a_k}\,)_+^2\} \tag{A1.9}$$

APPENDIX II.

Proof of Theorem 2: The colored noise minimax problem with n observations is to minimize J_n where

$$J_n(B_n) \overset{\Delta}{=} \max_{\vec{X}_n^* \Lambda_n \vec{X}_n \le Pn} \frac{1}{n} \{|(I-B_n)\vec{X}_n|^2 + \mathrm{Tr}(B_n^* B_n U_n^* R_n U_n)\}$$

$$= \max_{\vec{Y}_n^* \Lambda_n \vec{Y}_n \le P} \{|(I-B_n)\vec{Y}_n|^2\} + \frac{1}{n} \mathrm{Tr}(B_n^* B_n U_n^* R_n U_n) \qquad (A2.1)$$

where $\vec{Y}_n \overset{\Delta}{=} \frac{1}{\sqrt{n}} \vec{X}_n$ and $\Lambda_n = \mathrm{diag}(a_0^n, a_1^n, \ldots, a_{n-1}^n)$ with $a_k^n = |G(\frac{k}{n})|^2 \le M_1^2$. U_n is the appropriate n dimensional exponential basis matrix, and R_n is the covariance matrix of the n noise random variables. Clearly (A2.1) is a restatement of (A1.7) making dependence on sample size explicit.

Define $\mathcal{E}$ to be the asymptotic minimax error, i.e.,

$$\mathcal{E} \overset{\Delta}{=} \lim_{n\to\infty} \min J_n \qquad (A2.2)$$

To find (A2.2), the strategy is to replace (A2.1) with a simplified problem whose solution is readily obtained, and show equality in the limit. With this in mind define

$$S_n \overset{\Delta}{=} \mathrm{diag}(S(\tfrac{k}{n})) \qquad k = 0, 1, \ldots, n-1 \qquad (A2.3)$$

and consider the alternative problem: minimize $\tilde{J}_n$ where

$$\tilde{J}_n(B_n) \overset{\Delta}{=} \max_{\vec{Y}_n^* \Lambda_n \vec{Y}_n \le P} |(I-B_n)\vec{Y}_n|^2 + \frac{1}{n} \mathrm{Tr}(B_n^* B_n S_n) \,. \qquad (A2.4)$$

The asymptotic minimax error, $\lim_{n\to\infty} \min \tilde{J}_n$, is easily found by an application of the previously stated Lemma 1. By Lemma 1, there exists matrices $\{D_n\}$ such that for each n

$$\tilde{J}_n(D_n) = \min_B \tilde{J}_n(B) \qquad (A2.5)$$

$$D_n = \mathrm{diag}((1 - \gamma_n^{1/2}|G(\tfrac{k}{n})|)_+) \qquad k = 0, 1, \ldots, n-1 \qquad (A2.6)$$
$$\text{for some } \gamma_n > 0$$

Thus, the minimax error is

$$\hat{\mathcal{E}}_n = \min_{\gamma_n} \{\gamma_n P + \frac{1}{n} \sum_{k=0}^{n-1} S(\tfrac{k}{n})(1 - \gamma_n^{1/2}|G(\tfrac{k}{n})|)_+^2\} \qquad (A2.7)$$

For large n, the sum is a Riemann sum approximating a well defined integral, hence the asymptotic minimax error is obtained by solving

$$E \triangleq \min_{\gamma} \{\gamma P + \int_0^1 S(x)(1 - \gamma^{1/2}|G(x)|)_+^2 \, dx\} \qquad (A2.8)$$

Heuristically speaking to obtain the asymptotic error in colored noise, it is only necessary to observe that the Topelitz matrix, R_n, "looks like" a circulant matrix for large n. In this case S_n acts like $U_n^* R_n U_n$ (in a sense to be made rigorous in lemma 2) and consequently, $\mathrm{Tr}(B_n^* B_n S_n) \approx \mathrm{Tr}(B_n^* B_n U_n^* R_n U_n)$ and we see that the two error expressions J_n and $\tilde{J}_n$ will be asymptotically equal.

Proceeding, let $\check{B}_n$ be the $n \times n$ (complex) matrix which minimizes J_n. So, for each n we have

$$J_n(D_n) \geq J_n(\check{B}_n) \qquad (A2.9)$$

where D_n is the minimizer of $\tilde{J}_n$. It is easy to see that the $\{\check{B}_n\}$ are uniformly bounded in norm, i.e. there exists constant M such that $\|\check{B}_n\| < M$, for every n, where $\|\check{B}_n\| \triangleq \sup_{x \neq 0} |\check{B}_n x|/|x|$.

To find the limit of $J_n(\check{B}_n)$ we need the following lemma.

Lemma 2: Let $S(x)$ be a continuous spectral density for a discrete random process defined on the integers. Let R_n designate the correlation matrix for n samples. Denote the DFT basis as

$$U_n \triangleq \left(\left(\frac{e^{\frac{-2\pi ikm}{n}}}{\sqrt{n}}\right)\right) .$$

With $\{B_n\}$ a sequence of $n \times n$ complex matrices uniformly bounded in norm, we have

$$\left|\frac{1}{n} \mathrm{Tr}(B_n^* B_n U_n^* R_n U_n) - \frac{1}{n} \mathrm{Tr}(B_n^* B_n S_n)\right| \xrightarrow[n\to\infty]{} 0$$

where $\qquad S_n \triangleq \mathrm{diag}(S(\frac{k}{n})) \qquad k = 0, 1, \ldots, n-1 .$

Proof: See Appendix III.

As previously shown $\tilde{J}_n(D_n) \to E$ and by Lemma 2

$$|J_n(D_n) - \tilde{J}_n(D_n)| = \left|\frac{1}{n} \mathrm{Tr}(D_n^* D_n U_n^* R_n U_n) - \frac{1}{n} \mathrm{Tr}(D_n^* D_n S_n)\right| \to 0 \qquad (A2.10)$$

Therefore $J_n(D_n) \to E$ and

$$E \geq \overline{\lim}\, J_n(\check{B}_n) \geq \underline{\lim}\, J_n(\check{B}_n) \geq 0. \tag{A2.11}$$

To conclude, consider a subsequence $\{\check{B}_{n(k)}\}$ such that $J_{n(k)}(\check{B}_{n(k)}) \to \underline{\lim}\, J_n(\check{B}_n)$, applying Lemma 1 and evoking the minimality of D_n we have

$$\underline{\lim}\, J_n(\check{B}_n) = \lim J_{n(k)}(\check{B}_{n(k)}) \geq E \tag{A2.12}$$

and hence

$$\lim J_n(\check{B}_n) = E \tag{A2.13}$$

APPENDIX III

Length considerations prohibit inclusion of the proof of Lemma 2. This proof involves first "windowing" the noise correlation matrix using the Fejér kernel and then approximating the resulting Toeplitz matrix by a circulant matrix. The asymptotic behavior of the desired quantity, $\mathrm{Tr}(B_n^* B_n U_n^* R_n U_n)$, may be derived. Toeplitz matrix approximation by circulants technique is discussed in a tutorial paper by R. M. Gray [10]. A more general though more difficult mathematical treatment may be obtained from Grenander and Szegö [11].

References

[1] G. Wahba, "Smoothing noisy data with spline functions,"Numerical Mathematics, Vol. 24, pp. 383-393, 1975.

[2] A. Demmler and C. Reinsch, "Oscillation matrices with spline smoothing," Numerical Mathematics, Vol. 24, pp. 375-382, 1975.

[3] M. Nussbaum, "Exact asymptotic minimax results for integrated mean square error in nonparametric regression," to appear in 1985 Annals of Statistics.

[4] P. Speckman, "Spline smoothing and optimal rates of convergence in nonparametric regression models," to appear in 1985 Annals of Statistics.

[5] M. S. Pinsker, "Optimal filtration of square-integrable signals in Gaussian noise," Problems in Information Transmission, Vol. 16, No. 2, pp. 120-133, April-June 1980.

[6] E. M. Glaser and D. S. Ruchkin, Principles of Neurobiological Signal Analysis. New York: Academic Press, 1976.

[7] J. P. C. de Weerd and W. L. J. Martens, "Theory and practice of a posteriori "Wiener" filtering of average evoked potentials," Biological Cybernetics, Vol. 30, pp. 81-94, 1978.

[8] J. P. C. de Weerd, "A posterior time varying filtering of averaged evoked potentials: I. Introduction and conceptual basis," Biological Cybernetics, Vol. 41, pp. 211-222, 1981.

[9] P. J. Sherman, W. P. Birkemeier, and J. P. de Weerd "On the optimal estimation of an unknown deterministic signal vector using a time-invariant filter," to appear in Acoustics, Speech and Signal Processing.

[10] R. M. Gray, "On the asymptotic eigenvalue distribution of Toeplitz matrices," IEEE Transactions on Information Theory, Vol. IT-18, No. 6, pp. 725-730, November 1972.

[11] U. Grenander and G. Szegö, Toeplitz Forms and Their Applications. New York: Chelsea Publishing Company, 1958.

A WELL-POSEDNESS PROPERTY OF A CLASS OF VARIATIONAL PROBLEMS AND ITS APPLICATION TO NONLINEAR ESTIMATION

JOHN M. MORRISON
Department of Mathematics
University of Texas at Austin
Austin, Texas 78712

and

GARY L. WISE
Departments of Electrical and Computer Engineering and Mathematics
University of Texas at Austin
Austin, Texas 78712

ABSTRACT

In this paper we consider a class of nonlinear estimators that are the solutions to a certain variational problem. These estimators generalize the concept of conditional expectation; we investigate their continuity and convergence properties.

I. INTRODUCTION

In this paper we will develop some machinery that generalizes some well known approximation and continuity properties of minimum mean square error analysis. These will allow us to examine estimation theory in a much more general context and give us a much wider choice of criteria for analyzing and penalizing error. We will be required to forego some of the main convenient accoutrements of minimum mean square error analysis: the Hilbert space structure of L^2, the projection theorem, and the fact that minimum norm projection onto a closed subspace is a nonexpansive linear operator. However, in return we receive a method that allows us to customize our notion of error to our particular model. Oftentimes, mean square error is not the appropriate fidelity criterion.

II. PRELIMINARIES

Throughout this paper let $(\Omega,\mathscr{S},P)$ denote a probability space and let $L^0(\Omega,\mathscr{S},P)$ denote the set of all random variables $X:\Omega\to\mathbb{R}$ modulo a.s. equivalence equipped with the topology of convergence in probability.

Let $\Phi: [0,\infty)\to[0,\infty)$ be convex, increasing, and satisfy $\Phi(0)=0$. We define the Orlicz space

$$L^\Phi(\Omega,\mathscr{S},P) = \{X\in L^0(\Omega,\mathscr{S},P): \int_\Omega \Phi(|X|)dP < \infty\}.$$

The function Φ is said to satisfy the doubling condition if there exist $C,M>0$ such that $x\geq M \Rightarrow \Phi(2x)\leq C\Phi(x)$. If Φ satisfies the doubling condition, $L^\Phi(\Omega,\mathscr{S},P)$ is a vector space. Moreover, it is a Banach space with the Luxemburg norm

$$||X||_{L^\Phi} = \inf\{\lambda>0: \int_\Omega \Phi\left(\frac{|X|}{\lambda}\right)dP \leq \Phi(1)\}.$$

Note that this is simply the Minkowski functional of a certain subset of L^Φ. Also, if $\{X_n\}_{n=1}^\infty$ and X are in L^Φ, $X_n \xrightarrow{L^\Phi} X$ if and only if $\lim_{n\to\infty}\int_\Omega \Phi(|X_n-X|)dP = 0$.

For the basic facts on Orlicz spaces see [3]. Details on Minkowski functions can be found in [1], pp. 294-295 or [6], pp. 23-26.

Let $\Phi: [0,\infty)\to[0,\infty)$ be increasing and have a strictly increasing first derivative ϕ with $\phi(x)\to\infty$ as $x\to\infty$. Since $\phi: [0,\infty)\to[0,\infty)$ is a homeomorphism, $\psi = \phi^{-1}$ is also continuous, strictly increasing, and satisfies $\lim_{x\to\infty} \psi(x) = \infty$. We define the conjugate function Ψ to Φ to be

$$\Psi(x) = \int_0^x \psi(t)dt, \quad x \geq 0.$$

This is somewhat more restrictive than the definition employed in [3], but it eliminates certain annoying technicalities. It also allows us to dispense with the customary nonatomicity assumption made about the measure spaces used in this context. We call the pair Φ,Ψ an Orlicz pair.

The Banach space L^Φ is reflexive if and only if Ψ and Φ satisfy the doubling condition. A simple calculation shows that the strict convexity of Φ implies that of the Luxemburg norm on L^Φ and that of the convex functional $X \to \int_\Omega \Phi(|X|)dP$. In addition, if the Banach space L^Φ is reflexive, we have $L^{\Phi*} \cong L^\Psi$ [3]. Hereafter we posit all Orlicz spaces are reflexive.

Let B be a Banach space. We say B is (i) locally uniformly convex if whenever $\{x_n\}_{n=1}^\infty$ and $\{y_n\}_{n=1}^\infty$ are sequences in the unit ball of B with $\lim_{n\to\infty} ||x_n+y_n|| = 2$, $\lim_{n\to\infty}||x_n-y_n|| = 0$; (ii) uniformly convex if $\forall\varepsilon > 0$ there exists $\delta > 0$ so that for any $x,y\epsilon B$ so that $||x|| = ||y|| = 1$, $||x+y|| \geq 2(1-\delta) \Rightarrow ||x-y|| < \varepsilon$. Uniform convexity is a much stronger condition than local uniform convexity. Every uniformly convex space is reflexive [6], pp. 126-128. For more about these sorts of conditions, see [2].

If B is a Banach space, $Q \subseteq B$ is said to be proximal if $\forall x\epsilon B$, the problem

$$\min_{z\epsilon Q} ||x-z||$$

possesses a solution. If B is reflexive we may apply the Smul'lyan theorems to see that every closed convex subset is proximal. Moreover, if B is strictly convex the norm minimizer problem has at most one solution. In this case for $x\epsilon B$ and $K \subseteq B$ that is closed and convex let $p_K(x)$ denote the unique solution to

$$\min_{z\epsilon K} ||x-z||.$$

The map p_K is called the metric projection of B onto K. Shortly we will see that if B is locally uniformly convex and reflexive then for any closed convex $K \subseteq B$, p_K is norm continuous. In [5] we showed that p_K is weakly sequentially continuous if B is reflexive and strictly convex.

Let M be a separable metric space. A Borel measurable map $Q: M \to M$ is said to be a round off map if Q has finite range, say $\{p_1,\ldots,p_n\}$ = range Q and $Q(p_k) = p_k$, $1 \leq k \leq n$. The set $\{Q^{-1}(p_1),\ldots,Q^{-1}(p_n)\}$ is called the partition of M defined by Q. A sequence $\{Q_n\}_{n=1}^\infty$ of round off maps is called a round off scheme if (i) $\forall x\epsilon M$, $\lim_{n\to\infty} \text{dia}\ Q_n^{-1}(Q_n(x)) = 0$; (ii) the partition of M defined by Q_{n+1} refines that defined by Q_n, $n\epsilon\mathbf{N}$. Note $\sigma(Q_n) \subseteq \sigma(Q_{n+1})$, $n\epsilon\mathbf{N}$. See [5] for details on round off schemes.

The primary result of [5] is contained in the

Theorem 1: Let Φ,Ψ be an Orlicz pair, $(\Omega,\mathscr{S},P)$ be a nonatomic probability space, M be a separable metric space and $\{X(t)\colon t\epsilon[0,T]\}$ be a stochastically continuous process on $(\Omega,\mathscr{S},P)$ taking values in M. Then for any round off scheme $\{Q_n\}_{n=1}^\infty$ on M and any increasing sequence $\{P_m\}_{m=1}^\infty$ of partitions of $[0,T]$ whose meshes decrease to zero we have for any $Y \epsilon L^\Phi(\Omega,\mathscr{S},P)$

$$\| E_{\Phi}(Y|Q_n(X(t)): t\epsilon P_m) - E_{\Phi}(Y|X(t): t\epsilon[0,T])\|_{L^{\Phi}} \to 0$$

as m,n →∞; or, equivalently,

$$0 = \lim_{m,n\to\infty} \int_{\Omega} \Phi(|E_{\Phi}(Y|Q_n(X(t)): t\epsilon P_m) - E_{\Phi}(Y|X(t): t\epsilon[0,T])|)dP$$

where $E_{\Phi}(\cdot|\mathscr{F}): L^{\Phi}(\Omega,\mathscr{S},P) \to L^{\Phi}(\Omega,\mathscr{F},P)$ denotes metric projection and $\mathscr{F}$ is any σ-subalgebra of $\mathscr{S}$.

III. A CONVEXITY CONDITION FOR CONVEX FUNCTIONALS ON REFLEXIVE BANACH SPACES

Let B be a reflexive Banach space and $\psi: B \to \mathbb{R}$ be a convex, norm continuous map satisfying

$$\lim_{\|x\|\to\infty} \psi(x) = +\infty .$$

Then using the Smul'lyan theorems we see that $\forall x \epsilon B$

$$\min_{z\epsilon K} \psi(x-z) \tag{1}$$

possesses a closed convex nonvoid subset L_x of K of solutions. Furthermore, if ψ is strictly convex each L_x is a singleton $\{p(x)\}$. We call the map $p: B \to K$ the solution map of (1). Let $\delta > 0$; define

$$K_{\delta,x} = \{z\epsilon K: \psi(x-z) < \delta + \psi(x-p(x))\}.$$

We say ψ is locally uniformly convex if $\lim_{\delta\downarrow 0} \text{dia } K_{\delta,x} = 0$ for all $x\epsilon B$ and closed convex $K \subseteq B$.

Proposition 2: If $\psi: B \to \mathbb{R}$ is locally uniformly convex and uniformly continuous on bounded subsets of B, $K \subseteq B$ is nonvoid convex and closed and $p: B \to K$ is the solution map of (1) then p is norm continuous.

Proof: Let $x_n \to x$ in B and $\delta > 0$. Take $z\epsilon K_{\delta,x}$. Then

$$\psi(x-z) < \delta + \psi(x-p(x)).$$

Since $x_n \to x$ there exists $N\epsilon\mathbb{N}$ so that $n \geq N \Rightarrow$

$$\psi(x_n-z) < \delta + \psi(x_n-p(x)).$$

By minimality

$$\begin{aligned} \psi(x_n-p(x_n)) &\leq \psi(x_n-z) \\ &< \delta + \psi(x_n-p(x)). \end{aligned}$$

Taking limit suprema,

$$\limsup_{n\to\infty} \psi(x_n-p(x_n)) \leq \psi(x-p(x)). \tag{2}$$

Suppose by way of the contrapositive that $\{p(x_n)\}_{n=1}^{\infty}$ is unbounded. Since $x_n \to x$, $\{x_n\}_{n=1}^{\infty}$ is bounded; this forces $\{x_n-p(x_n)\}_{n=1}^{\infty}$ to be unbounded. Observing that $\lim_{\|x\|\to\infty} \psi(x) = +\infty$ it now follows that

$$\limsup_{n\to\infty} \psi(x_n-p(x_n)) = +\infty,$$

a flagrant violation of (2). This insures that $\{p(x_n)\}_{n=1}^{\infty}$ is bounded.

Suppose that for some $z_0\epsilon B$, $p(x_n) \rightharpoonup z_0$. Then $x_n - p(x_n) \rightharpoonup x-z_0$; ψ is convex and norm continuous and therefore weakly lowersemicontinuous. Thus

$$\psi(x-z_0) \le \liminf_{n\to\infty} \psi(x_n-p(x_n))$$
$$\le \limsup_{n\to\infty} \psi(x_n-p(x_n)) \le \psi(x-p(x)).$$

Next observe that K is closed and convex and therefore weakly closed. Thus $p(x_n)\in K$, $n\in\mathbb{N}$, $p(x_n)\rightharpoonup z_0 \Rightarrow z_0\in K$. By minimality, $z_0 = p(x)$.

Since B is reflexive and $\{p(x_n)\}_{n=1}^{\infty}$ is bounded each subsequence of $\{p(x_n)\}_{n=1}^{\infty}$ must have a further weakly convergent subsequence. But the foregoing discussion tells us that this further subsequence must converge to $p(x)$. Immediately it follows $p(x_n) \rightharpoonup p(x)$.

We may apply the weak lower semicontinuity of ψ to see that

$$\psi(x-p(x)) \le \liminf_{n\to\infty} \psi(x_n-p(x_n)).$$

Combine this with (2) to get

$$\psi(x-p(x)) = \lim_{n\to\infty} \psi(x_n-p(x_n)).$$

Put $M = 1+\sup_n \|x_n-p(x_n)\|$; note $M<\infty$. Take $\varepsilon>0$, choose $\eta>0$ s.t. $\eta<1$ and $\|x-y\|<\eta$, $\|x\|,\|y\| \le M \Rightarrow |\psi(x)-\psi(y)|<\varepsilon$. Because $x_n\to x$ there exists $N\in\mathbb{N}$ such that $n\ge N \Rightarrow \|x_n-x\|<\eta$. Then $n\ge N \Rightarrow$

$$\|x_n-p(x_n)-(x-p(x_n))\| = \|x_n-x\| < \eta$$

so $n\ge N \Rightarrow$

$$|\psi(x_n-p(x_n)) - \psi(x-p(x))| < \varepsilon.$$

This forces $\lim_{n\to\infty}\{\psi(x_n-p(x_n))-\psi(x-p(x_n))\} = 0$ and hence $\lim_{n\to\infty} \psi(x-p(x_n)) = \psi(x-p(x))$.

Noting $\lim_{n\to\infty} \psi(x-p(x_n)) = \psi(x-p(x))$ there exists $N\in\mathbb{N}$ so that $n\ge N \Rightarrow$

$$\psi(x-p(x_n)) < \delta+\psi(x-p(x)).$$

But this says that for $n\ge N$, $p(x_n)\in K_{\delta,x}$. Obviously $p(x)\in K_{\delta,x}$ so for $n\ge N$, $\|p(x_n)-p(x)\| \le \operatorname{dia} K_{\delta,x}$. Since $\lim_{\delta\to0} \operatorname{dia} K_{\delta,x} = 0$ we must have $p(x_n)\to p(x)$. QED

Proposition 3: Suppose $\{K_n\}_{n=1}^{\infty}$, K are nonvoid closed convex subsets of B with $K_n \subseteq K_{n+1}$, $n\in\mathbb{N}$ and $K = \overline{\bigcup_{n=1}^{\infty} K_n}$. Let $\psi: B\to\mathbb{R}$ be locally uniformly convex and norm continuous. Finally denote by p_n the solution map of (1) corresponding to K_n, $n\in\mathbb{N}$ and p denote the solution map of K. Then for $x\in B$, $p_n(x)\to p(x)$.

Proof: Let $\delta>0$, $x\in B$. By the continuity of ψ, $K_{\delta,x}$ is a relative neighborhood of $p(x)$ in K. Noting $K = \overline{\bigcup_{n=1}^{\infty} K_n}$, we see there must be an $N\in\mathbb{N}$ so that $n\ge N \Rightarrow K_n \cap K_{\delta,x} \ne \emptyset$.

Fix $n\ge N$, $z\in K_{\delta,x}\cap K_n$. Then we know that (i) $\psi(x-z) < \delta+\psi(x-p(x))$; (ii) $\psi(x-p_n(x)) \le \psi(x-z)$. Assembling the facts, $n\ge N \Rightarrow$

$$\psi(x-p_n(x)) \le \delta+\psi(x-p(x)).$$

Letting $n\to\infty$,

$$\limsup_{n\to\infty} \psi(x-p_n(x)) \le \psi(x-p(x)). \qquad (3)$$

We may imitate a familiar argument in Proposition 2 to see that $\{p_n(x)\}_{n=1}^{\infty}$ is bounded.

Since $K_n \subseteq K_{n+1} \subseteq K$, $n\in\mathbb{N}$ it is easy to see that

$$\psi(x-p_n(x)) \ge \psi(x-p_{n+1}(x)) \ge \psi(x-p(x)).$$

With the assistance of (3) we obtain

$$\psi(x-p(x)) = \lim_{n\to\infty} \psi(x-p_n(x)).$$

Pick $\delta > 0$. There exists $N\in\mathbb{N}$ such that $n \ge N \Rightarrow$

$$\psi(x-p_n(x)) < \psi(x-p(x)) + \delta.$$

We may now conclude $p_n(x)\in K_{\delta,x}$, $n \ge N$. Since dia $K_{\delta,x} \to 0$ as $\delta \to 0$ we conclude that $p_n(x) \to p(x)$ in norm. QED

IV. SOME FACTS ABOUT ORLICZ SPACES

Theorem 4: Let $(\Omega,\mathscr{S},P)$ be a probability space and Φ,Ψ be an Orlicz pair. Then $L^{\Phi}(\Omega,\mathscr{S},P)$ is uniformly convex if and only if

$\forall\, \varepsilon$ such that $0 < \varepsilon < \frac{1}{4}$ there exists $R_{\varepsilon} > 0$ so that

$$\liminf_{n\to\infty} \frac{\psi(x)}{\psi((1-\varepsilon)x)} > R_{\varepsilon}.$$

Proof: This is the central result of [4].

Define $\xi\colon L^{\Phi}(\Omega,\mathscr{S},P) \to [0,\infty)$ by $\xi(X) = \int_{\Omega} \Phi(|X|)\,dP$. Then ξ is a strictly convex norm continuous functional and $\lim_{\|X\|_{L^\Phi}\to\infty} \xi(X) = +\infty$. Let $0 < \varepsilon < 1$, $X,Y \in L(\Omega,\mathscr{S},P)$ and $X \ne Y$ a.s.[P]. Then $\|X-Y\|_{L^\Phi} \ne 0$. Using the convexity of Φ,

$$\begin{aligned}\Phi(|X-Y|) &= \Phi(\varepsilon(\tfrac{1}{\varepsilon}|X-Y|) + (1-\varepsilon)(0)) \\ &\le \varepsilon\,\Phi(\tfrac{|X-Y|}{\varepsilon}) + (1-\varepsilon)\Phi(0) \\ &= \varepsilon\,\Phi(\tfrac{|X-Y|}{\varepsilon}).\end{aligned}$$

Integrating,

$$\int_{\Omega}\Phi(|X-Y|)\,dP \le \varepsilon \int_{\Omega}\Phi(\tfrac{|X-Y|}{\varepsilon})\,dP.$$

Now suppose $\|X-Y\|_{L^\Phi} \le 1$. Then

$$\begin{aligned}\int_{\Omega}\Phi(|X-Y|)\,dP &\le \|X-Y\|_{L^\Phi} \int_{\Omega}\Phi\Big(\frac{|X-Y|}{\|X-Y\|_{L^\Phi}}\Big)\,dP \\ &= \|X-Y\|_{L^\Phi}\,\Phi(1),\end{aligned}$$

amply demonstrating the global uniform continuity of ξ.

We will not address the question of the local uniform convexity of ξ here. The hypothesis of Theorem 4 or possibly something similar might guarantee the local uniform convexity of ξ.

V. A CONVERGENCE PRINCIPLE FOR A GENERALIZED CONDITIONAL EXPECTATION

Definition: Let $(\Omega,\mathscr{S},P)$ be a probability space, $\mathscr{F}$ be a sub σ-algebra of $\mathscr{S}$ and Φ,Ψ be an Orlicz pair. For $X\in L^\Phi(\Omega,\mathscr{S},P)$, we define $\hat{E}_\Phi(X|\mathscr{F})$ to be the unique solution to the variational problem

$$\min\left\{\int_\Omega \Phi(|X-Z|)dP: Z\in L^\Phi(\Omega,\mathscr{F},P)\right\}.$$

Recall from [5] that if $\Phi(x) = \frac{x^2}{2}$ for $x \geq 0$, then $\hat{E}_\Phi(\cdot|\mathscr{F})$ is ordinary conditional expectation given $\mathscr{F}$.

Remark: If the functional $X \overset{\xi}{\longmapsto} \int_\Omega \Phi(|X|)dP$ is locally uniformly convex then $\hat{E}_\Phi(\cdot|\mathscr{F})$ is norm continuous on $L^\Phi(\Omega,\mathscr{S},P)$.

Theorem 5: Let $(\Omega,\mathscr{S},P)$ be a probability space, $\{\mathscr{F}_n\}_{n=1}^\infty$ be an increasing family of sub σ-algebras of $\mathscr{S}$ and put $\mathscr{F} = \bigvee_{n=1}^\infty \mathscr{F}_n$. Suppose Φ,Ψ is an Orlicz pair so that $X \rightarrow \int_\Omega \Phi(|X|)dP$ is locally uniformly convex. Then for $X\in L^\Phi(\Omega,\mathscr{S},P)$,

$$\hat{E}_\Phi(X|\mathscr{F}_n) \xrightarrow{L^\Phi} \hat{E}_\Phi(X|\mathscr{F}) \text{ as } n\rightarrow\infty,$$

or equivalently,

$$\lim_{n\rightarrow\infty} \int_\Omega \Phi(|\hat{E}_\Phi(X|\mathscr{F}_n)-\hat{E}_\Phi(X|\mathscr{F})|)dP = 0.$$

Proof: This is an immediate consequence of Proposition 3. QED

We may now proceed as in [5] to get

Theorem 6: Let $(\Omega,\mathscr{S},P)$ be a probability space, $\{X(t): t\in[0,T]\}$ be a process on $(\Omega,\mathscr{S},P)$ taking values in a separable metric space M and Φ,Ψ be an Orlicz pair with $X \overset{\xi}{\longmapsto} \int_\Omega \Phi(|X|)dP$ locally uniformly convex. Then if $\{Q_n\}_{n=1}^\infty$ is a round off scheme for M and $\{P_m\}_{m=1}^\infty$ is an increasing sequence of partitions of $[0,T]$ whose meshes decrease to zero, we have for $Y\in L^\Phi(\Omega,\mathscr{S},P)$

$$\hat{E}_\Phi(Y|Q_n(X(t)): t\in P_m) \xrightarrow{L^\Phi} \hat{E}_\Phi(Y|X(t): t\in[0,T])$$

as $m,n\rightarrow\infty$, or equivalently,

$$\lim_{m,n\rightarrow\infty} \int_\Omega \Phi(|\hat{E}_\Phi(Y|Q_n(X(t)): t\in P_m) - \hat{E}_\Phi(Y|X(t): t\in[0,T])|)dP = 0.$$

VI. CONCLUSIONS AND A PARTING SHOT AT EARLIER WORK

The abstract principle developed in Section III subsumes the Banach space principle developed in [5]. Simply take the functional to be the norm.

We have enhanced the feasibility of studying the operators $\hat{E}_\Phi$ numerically in two ways. First, knowing that $\hat{E}_\Phi$ is continuous assures us that it will tolerate small L^{Φ}-perturbations. Second, the result of Theorem 6 shows us we may approximate $\hat{E}_{\Phi}(Y|X(t): t\epsilon[0,T])$ by $\hat{E}_{\Phi}(Y|Q_n(X(t)): t\epsilon P_m)$. Since $\sigma(Q_n(X(t)): t\epsilon P_m)$ is finite, calculation of $\hat{E}_{\Phi}(Y|Q_n(X(t)): t\epsilon P_m)$ consists of solving a finite dimensional nonlinear optimization problem. A large body of knowledge now exists about the numerical solution of such problems.

ACKNOWLEDGEMENTS

This research was supported by the Air Force Office of Scientific Research under Grants AFOSR-81-0047 and its successor. The authors would like to thank the Banach Space and Approximation Theory groups at the University of Texas at Austin for many helpful suggestions.

REFERENCES

1. Brown, A., and Pearcy, C., *Introduction to Operator Theory I*. New York: Springer-Verlag, 1977.
2. Day, M., *Normed Linear Spaces*. New York: Springer Verlag, 1973.
3. Krosnoselskii, M., and Rutickii, Y., *Convex Functions and Orlicz Spaces*. Groningen: P. Noordhoff, 1961.
4. Milnes, W., "Convexity of Orlicz Spaces," *Pacific Journal of Mathematics*, Vol. 7, No. 3, 1957, pp. 1451-1483.
5. Morrison, J., and Wise, G., "A Stability Property of Conditional Expectations," *Proc. Nineteenth Annual Conference on Information Sciences and Systems*, Baltimore, MD, March 27-29, 1985, pp. 226-229.
6. Yosida, K., *Functional Analysis*, 6th edition. New York: Springer-Verlag, 1980.
7. Zeidler, E., *Nonlinear Functional Analysis*, Vol. III. New York: Springer-Verlag, 1985.

Signal-to-Noise Ratio and Central Limit Theorem Considerations in Non-Gaussian Detection

Douglas Warren and John B. Thomas
Department of Electrical Engineering
Princeton University
Princeton, New Jersey 08544

ABSTRACT

A generalized signal-to-noise ratio for detectors is defined and its relationship to quality of detection is investigated. Specific attention is given to log-likelihood ratio detectors. The question is discussed as to whether a Central Limit Theorem approximation for the output of the detector gives results which are close to those of the actual detector and is illustrated with examples.

I. INTRODUCTION

In this paper we will be looking at the discrete time problem of detection of a constant signal in the presence of independent, identically distributed noise; that is,

$$\text{Under } H: \; X_i = N_i \quad i = 1,2,\ldots,n$$

and

$$\text{Under } K: \; X_i = N_i + s \quad i = 1,2,\ldots,n$$

where X_i is the observation, N_i is the noise and s is a positive constant. The only limitations that we will place on the possible noise distributions will be that they are symmetric, bounded, unimodal and have zero means.

If the detector is linear, we can define a signal-to-noise ratio (SNR) in terms of s, n, and the noise variance σ^2. In this case the SNR is clearly a useful quantity: for a constant false alarm probability the probability of detection is a monotonic function of the SNR. If the detector is nonlinear, not only is the relationship between SNR and detector performance less clear but it is no longer immediately apparent what definition of SNR would be useful.

A generalized SNR has been proposed and it has been shown to be maximized when the nonlinearity is a monotonic function of the likelihood ratio[1,2]. This is an encouraging result since any monotonic function of the likelihood ratio is also the Neyman-Pearson optimal detector. Following some unpublished work of Poor, Chiang and Wise[3], we show later that, if the sample size is very large, this SNR functions much as does the conventional SNR for the linear detector. In fact, it can be shown that, asymptotically, the relationship between SNR and the probability of detection is once again monotonic.

As with any asymptotic result there arises here the basic problem of determining whether the Central Limit Theorem holds and the more obscure question of deciding how many samples are needed before the implications of the theorem merge with reality. Obviously, the theorem is useless as a practical tool if it is necessary to collect an infinite number of samples before it can be invoked and as a result it is desirable to know just how much is enough. In what follows we show that, under some mild restrictions, the output of a log-likelihood ratio detector obeys the Central Limit Theorem and we look at some examples in order to illustrate the rate of convergence.

The paper is composed of three parts and an appendix containing proofs. In Part I we define and discuss some relevant properties of the generalized SNR and the log-

---

This research was supported by the Office of Naval Research under Grant N00014-81-K-0146.

likelihood ratio. In Part II we investigate the relationship between SNR and traditional measures of detector performance. In Part III we look at some numerical examples concerning the convergence of the log-likelihood ratio to a Gaussian random variable. Part IV contains conclusions and a discussion of topics for further study.

I. PRELIMINARY PROPERTIES

The log-likelihood ratio. It is well known[4] that the likelihood ratio

$$L(\mathbf{X}) = \frac{f_K(\mathbf{X})}{f_H(\mathbf{X})} ,$$

maximizes the probability of detection β for a given false alarm rate α. Here f_K and f_H are the densities of the observation vector $\mathbf{X}$ under the hypotheses K and H respectively. If we are attempting to detect a shift to the right by a constant vector $\mathbf{s}$ and f is the density of the noise we have

$$L(\mathbf{X}) = \frac{f(\mathbf{X}-\mathbf{s})}{f(\mathbf{X})} .$$

It is also well known that any monotonically increasing function of the likelihood ratio will have the same receiver operating characteristic (ROC, that is, β vs. α) as that given by the likelihood ratio. The logarithm is, of course, one such monotonically increasing function. For the case of i.i.d. noise we have

$$L(\mathbf{X}) = \prod_{i=1}^{n} \frac{f(X_i - s)}{f(X_i)} .$$

By using $\log L$ rather than L we not only simplify our computation by converting the product to a sum

$$l(\mathbf{X}) = \log L(\mathbf{X}) = \sum_{i=1}^{n} \log \frac{f(X_i - s)}{f(X_i)} ,$$

but for large enough n we can approximate the test statistic $l(\mathbf{X})$ by using the Central Limit Theorem and thereby further simplify the problem.

In this paper we will discuss the log-likelihood ratio for unimodal, bounded, symmetric densities having zero means. We now list some properties which are relevant to this discussion:

1) The log-likelihood ratio is odd symmetric about $s/2$ [that is, $l(x) = log\,[f(x-s)/f(x)] = -l(s-x)$].
2) $E_H\, l(X) = -E_K\, l(X)$ where the subscripts indicate expectation under the given hypotheses.
3) $var_H\, l(X) = var_K\, l(X)$
4) Under certain mild restrictions, the variance of the test statistic $l(X)$ is finite.

Proofs:

1) $l(s-x) = \log \frac{f(x)}{f(s-x)} = \log \frac{f(x)}{f(x-s)} = -l(x)$ by the symmetry of f.
2) $E_H\, l(X) = \int l(X) f(X) dX = -\int l(s-X) f(X) dX = -\int l(Y) f(Y-s) dY = -E_K\, l(X)$
3) $E_H\, l^2(X) = \int l^2(X) f(X) dX = \int l^2(s-X) f(X) dX = \int l^2(Y) f(Y-s) dY = E_K\, l^2(X)$
4) proven in the Appendix.

Generalized signal-to-noise ratio (SNR). The motivation for defining a SNR for detectors is the same as that for doing so with any device; we want to know how strongly the output is affected by the signal and by the noise. For the linear detector the two effects are completely separable and the resulting SNR is only a function of the input SNR and the gain of the processor. For example, if the noise is Gaussian with zero mean and variance σ^2, then

$$\beta = \Phi\ [\Phi^{-1}(\alpha)+\frac{\sqrt{n}\ s}{\sigma}]$$

where Φ is the standard normal cumulative distribution function. Since the Φ function is monotonic in its argument it is clear that for fixed α, β is a monotonic function of $\sqrt{n}\ s/\sigma$. However, for nonlinear detectors we no longer have this separation property and the intuition which lends itself to the linear case is no longer evident. What is clear is that if we are to use the SNR as a measurement of detector performance then we want a higher SNR detector to have a higher probability of detection at a given false alarm rate than a detector with a lower SNR.

The following definition has been suggested[1] for a generalized SNR for detectors:

$$\mathrm{SNR}\ [g(X)] = \frac{E_K\ g(X) - E_H\ g(X)}{[\ var_H\ g(X)]^{1/2}}\ ,$$

where g is the processing non-linearity. The numerator represents the shift in the mean between the two hypotheses while the denominator represents the noise power at the output under the hypothesis H (we have shown earlier that the output variance is equal under both hypotheses for the log-likelihood detector). The definition has an intuitive feel to it and while it is not always monotonic in β for a fixed α we will show that, for a large enough sample size, it will be. Hence, asymptotically, this SNR is equivalent to the Neyman-Pearson criterion for quality of detection.

Some properties of the SNR[3].

1) SNR (k *samples*) = $\sqrt{k}$ × SNR (1 *sample*).

2) SNR is unaffected by gain or shift. That is,

$$\mathrm{SNR}\ [ag(X) + b\,] = \mathrm{SNR}\ [g(X)] \qquad a > 0.$$

3) For the log-likelihood ratio:

$$\mathrm{SNR}\ [l(X)] = \frac{2E_K\, l(X)}{[\ var\ l(X)]^{1/2}}\ .$$

II. SNR AND THE CENTRAL LIMIT THEOREM FOR LOG-LIKELIHOOD RATIOS

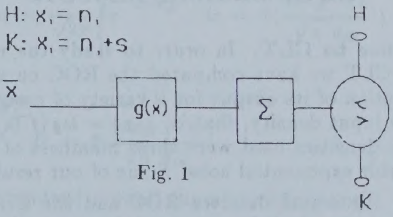


Fig. 1

The detector of Fig. (1) is in the form of a sum and hence if $log(f\ (X_i - s)/f\ (X_i))$ can be shown to have finite variance we can use the Central Limit Theorem to determine the asymptotic properties of $l(\mathbf{X})$. In this section we investigate the relationship between SNR and detector performance for sample sizes which are large enough so that the Central Limit Theorem can be invoked.

In [3] it is shown that if the two hypotheses are equally likely and $\alpha = 1-\beta$ then, asymptotically, the probability of error is monotonically decreasing in SNR. We generalize this statement with the following theorem.

Theorem 1: *Given a false alarm rate, α, a symmetric, bounded unimodal density, f , and a detector structure as shown in Fig. 1. Suppose we are comparing any detector g for which $var_K\ g(X) \geq var_H\ g(X)$ with the linear detector, $g(X) = X$, and suppose the sample size is large enough that the Central Limit Theorem can be invoked. Then a sufficient* *condition for higher probability of detection of a positive constant in noise with density* f *is higher signal-to-noise ratio. If* $var_K\, g(X) = var_H\, g(X)$ *then the condition is also necessary. Furthermore, if we are comparing two log-likelihood detectors and the Central Limit Theorem can be invoked then higher SNR is a necessary and sufficient condition for higher probability of detection.*

Proof (see Appendix A)

In [3] it is shown that for $P_H = P_K$ and $\alpha = 1-\beta$ the magnitude of the difference in the probability of error of the actual detector and the CLT approximation to it is $O(n^{-1/2})$ and that the relative error can actually grow exponentially in n. Here P_H and P_K are the prior probabilities under the two hypotheses. This is a somewhat discouraging result and elucidates the need for care when invoking limit theorems in finite sample problems. In Part III we will look at some graphical comparisons between the operating characteristics for some detectors and their CLT approximations.

The Central Limit Theorem and the log-likelihood ratio. We now must determine whether the log-likelihood ratio obeys the CLT so that we can apply the above theorem. We will show that for symmetric, unimodal and bounded densities having zero mean, if $\lim_{x \to \infty} g'(x) < \infty$, then the log-likelihood ratio has finite variance and the CLT holds. More formally we have the following theorem:

Theorem 2:

Suppose X *has the density* $f(x)$, *where* f *is symmetric, unimodal, bounded and has zero mean. Then the variance of the log-likelihood ratio* $g(x) = \log[f(x-s)/f(x)]$ *is finite if* $\lim_{x \to \infty} g'(x) < \infty$. *Furthermore, if* X *has density* h *where* h *is also symmetric, unimodal, bounded and has zero mean then* $g(x)$ *has finite variance if there exists* Y *such that* $h(y) \le k_1 e^{-k_2 y}$, $k_1, k_2 > 0$ *for all* $y > Y$.

Proof (see Appendix B)

III. NUMERICAL RESULTS

Rate of convergence to CLT. In order to study the rate of convergence to the results predicted by the CLT we have computed the ROC curves for the actual detector and for a CLT approximation of its output for a variety of cases. In all cases the detector was optimal for the noise input density, that is, $g(x) = \log[f(x-s)/f(x)]$ where f is the density of the noise. The densities used were three members of the Johnson System family[5] and Laplace, or double exponential noise. Some of our results are shown in Figs. 2-8.

Figs. 2 and 3 show the actual detector ROC and the CLT approximation to it for Johnson Noise with parameter $\delta=3$ for two and four samples respectively. There is a marked improvement in the quality of the approximation with the increase in the number of samples. Figs. 4 and 5 show the same quantities except δ was changed to 10. Again there is improvement although it is not as dramatic. It should be noted that in the limit, as $\delta \to \infty$, the distribution becomes normal. Hence we would expect the CLT approximation to improve as δ is increased. Fig. 6 shows the ROC and its approximation for $\delta=50$. It is obvious that the approximation in this case is very good.

In [8] a functional expression is given for the distribution of the output of the log-likelihood detector for Laplace Noise. From this expression we have calculated the ROC curves for this detector and the CLT approximation to it. The results are shown in Figs. 7 and 8 for five and ten samples, respectively. Here again we see improvement with an increase in the number of samples although it is not as large as in Figs. 1 and 2. This might be due to the fact that the log-likelihood ratio for Laplace noise places a finite mass at $\sqrt{2}s/\sigma$ which causes the output distribution to be discontinuous.

IV. CONCLUSIONS AND TOPICS FOR FURTHER STUDY

Conclusions. It is clear that in the limit SNR and the Neyman-Pearson criterion for detector performance are equivalent. The question of whether SNR is an appropriate measure for finite sample size detection is much murkier. As was mentioned, an upper bound on the magnitude of the error in predicting the probability of error is derived in [3]; however, as was demonstrated by the numerical examples, good results can be obtained in certain cases when the CLT is used, even in the case of very small sample size. From this we conclude that care should be taken when using SNR as a measure of detector performance because the accuracy of the resulting predictions is application dependent.

Topics for further study. In [6], Czarnecki designed a simple piece-wise linear processor for small-signal detection by maximizing the efficacy over a class of such processors. Owing to the similar functional forms of efficacy and SNR an analogous design might be produced by maximizing the SNR over some class of simple detectors.

Another course of action which might yield interesting results is attempting to find an error bound for CLT approximation for a given class of densities processed by a given class of detectors. Both the Johnson System[5] and the Generalized Gaussian[9] families would lend themselves to this endeavor.

APPENDIX A

Proof of Theorem 1: Given two detectors, g_1 and g_2. Let

$$m_{iJ} = \int g_i(X) f_J(X) dX \qquad i = 1,2 \qquad J = H, K \ ,$$

$$\sigma_{iJ}^2 = \int [g_i(X) - m_{iJ}]^2 f_J(X) dX \ ,$$

where $f_H(x) = f(x)$ and $f_K(x) = f(x-s)$. Since we assume the CLT holds, the density, under the hypothesis J, of the test statistic,

$$s_i(\mathbf{X}) = \sum_{m=1}^{n} g_i(X_m),$$

is normal with mean nm_{iJ} and variance $n\sigma_{iJ}^2$. We then have:

$$\alpha_i = \frac{1}{\sqrt{2\pi}} \int_{t_i}^{\infty} e^{\frac{(z-nm_{iH})^2}{2n\sigma_{iH}^2}} dz = \Phi[\frac{nm_{iH} - t_i}{\sqrt{n}\ \sigma_{iH}}]$$

and

$$\beta_i = \frac{1}{\sqrt{2\pi}} \int_{t_i}^{\infty} e^{\frac{(z-nm_{iK})^2}{2n\sigma_{iK}^2}} dz = \Phi[\frac{nm_{iK} - t_i}{\sqrt{n}\ \sigma_{iK}}] \tag{A.1}$$

where once again Φ is the standard normal cumulative function.

Equating the expressions for α_1 and α_2 and solving for t_2 yields:

$$t_2 = (t_1 - nm_{1H})\frac{\sigma_{2H}}{\sigma_{1H}} + nm_{2H} \ .$$

Substituting for t_2 in the expression for β_2 yields:

$$\beta_2 = \Phi\left(\frac{nm_{2K} - (t_1 - nm_{1H})\frac{\sigma_{2H}}{\sigma_{1H}} - nm_{2H}}{\sqrt{n}\ \sigma_{2K}} \right) \tag{A.2}$$

We now turn our attention to the case where g_1 is the linear detector; that is, $g_1(X) = X$. We then have:

$$m_{1K} = s \ , \ m_{1H} = 0, \ \sigma_{1H} = \sigma_{1K} = \sigma_1, \ t_2 = t_1\frac{\sigma_{2H}}{\sigma_1} + nm_{2H},$$

$$\beta_1 = \Phi\left(\frac{ns - t_1}{\sqrt{n}\ \sigma_1} \right) ,\beta_2 = \Phi\left(\frac{nm_{2K} - t_1 \dfrac{\sigma_{2H}}{\sigma_1} - nm_{2H}}{\sqrt{n}\ \sigma_{2K}} \right) .$$

Then $\beta_2 > \beta_1$ if and only if

$$\frac{nm_{2K} - t_1 \dfrac{\sigma_{2H}}{\sigma_1} - nm_{2H}}{\sigma_{2K}} > \frac{ns - t_1}{\sigma_1} . \tag{A.3}$$

After rearranging we have:

$$\text{SNR}_2 = \frac{m_{2K} - m_{2H}}{\sigma_{2H}} \geq \frac{m_{2K} - m_{2H}}{\sigma_{2K}} > \frac{s}{\sigma_1} - \frac{t_1}{n} \frac{(\sigma_{2K} - \sigma_{2H})}{\sigma_1 \sigma_{2K}} ,$$

where the second term on the right side is non-negative. Hence the condition

$$\text{SNR}_2 > \frac{s}{\sigma_1} \tag{A.4}$$

is sufficient for $\beta_2 > \beta_1$. If $\sigma_{2K} = \sigma_{2H}$ then Eq. (A.3) reduces to

$$\frac{m_{2K} - m_{2H}}{\sigma_{2H}} = \text{SNR}_2 > \frac{s}{\sigma_1} ,$$

and the condition in Eq. (A.4) is also necessary.

For the log likelihood ratio we have $\sigma_{iH} = \sigma_{iK}$. From Eqs. (A.1) and (A.2) we have $\beta_2 > \beta_1$ if and only if

$$\frac{nm_{2K} - (t_1 - nm_{1H}) \dfrac{\sigma_{2H}}{\sigma_{1H}} - nm_{2H}}{\sigma_{2K}} > \frac{nm_{1K} - t_1}{\sigma_{1K}} .$$

After some rearrangement we have

$$\frac{m_{2K} - m_{2H}}{\sigma_{2K}} > m_{1H} \frac{\sigma_{2H}}{\sigma_{1H} \sigma_{2K}} - \frac{t_1}{n} \left(\frac{1}{\sigma_{1K}} - \frac{\sigma_{2H}}{\sigma_{1H} \sigma_{2K}} \right) .$$

Under the equality constraint on the variances this reduces to

$$\text{SNR}_2 = \frac{m_{2k} - m_{2H}}{\sigma_2} > \frac{m_{1K} - m_{1H}}{\sigma_1} = \text{SNR}_1 .$$

APPENDIX B

Proof of Theorem 2: We are interested in showing that the following two integrals converge

$$E\ g^2 = \int_{-\infty}^{+\infty} g^2(X) f(X) dX \tag{B.1}$$

$$E\ g = \int_{-\infty}^{+\infty} g(X) f(X) dX \tag{B.2}$$

If $| g(x) | = | \log [f(x-s)/f(x)] |$ is bounded ($\leq M$) then we have

$$E\ g^2 = \int g^2 f \leq M^2 \int f = M^2$$

and

$$E\ g = \int g f \leq M \int f = M .$$

Hence for bounded g we have bounded variance.

We now look at the case where g is not bounded. In order to show that the integrals in Eqs. (B.1) and (B.2) converge we want to bound g and f with known functions for which convergence can be demonstrated. Since the problem is entirely symmetric we will concern ourselves only with integration from $+\frac{s}{2}$ to infinity. For a Gaussian density with variance σ^2

$$g_{Gau_\sigma}(x) = \frac{xs}{\sigma^2} - \frac{s^2}{2\sigma^2}$$

and for a double exponential density $f(x) = \gamma e^{-\gamma |x|}$ with variance $\sigma^2 = \frac{2}{\gamma^2}$

$$g_{Lap_\sigma} = \begin{cases} -\sqrt{2}\frac{s}{\sigma} & \text{if } x < 0 \\ \frac{\sqrt{2}}{\sigma}(2x-s) & \text{if } 0 \le x < s \\ \sqrt{2}\frac{s}{\sigma} & \text{if } x \ge s \end{cases}$$

Hence for any unbounded log-likelihood ratio with $\lim_{x \to \infty} g'(x) < \infty$ and for all $x \ge P$ we can say

$$g_{Lap_{\sigma_2}}(x) \le g(x) \le g_{Gau_{\sigma_1}}(x)\,, \tag{B.3}$$

where σ_1, σ_2 and P are positive constants. That is, g is bounded above by a straight line with positive slope and below by a positive constant. We then have

$$g(x) \le \frac{xs}{\sigma_1^2} - \frac{s^2}{2\sigma_1^2} \tag{B.4}$$

and

$$g(x) \ge \frac{\sqrt{2}s}{\sigma_2} \tag{B.5}$$

for all $x \ge P$. From Eq. (B.5) we have

$$\frac{f(x-s)}{f(x)} \ge e^{\frac{\sqrt{2}s}{\sigma_2}}$$

or

$$f(x) \le e^{\frac{-\sqrt{2}s}{\sigma_2}} f(x-s)\,. \tag{B.6}$$

We then have

$$f(s) \le e^{\frac{-\sqrt{2}s}{\sigma_2}} f(0)$$

$$f(2s) \le e^{\frac{-\sqrt{2}s}{\sigma_2}} f(s) \le e^{\frac{-2\sqrt{2}s}{\sigma_2}} f(0)$$

$$f(ks) \le e^{\frac{-\sqrt{2}s}{\sigma_2}} f[(k-1)s] \le e^{\frac{-k\sqrt{2}s}{\sigma_2}} f(0)$$

where due to the previously mentioned constraints on f we know it has a finite maximum at $x = 0$. We now define the step function f_1 as follows

$$f_1(x) = e^{\frac{-(k-1)\sqrt{2}s}{\sigma_2}} f(0) \qquad (k-1)s < x \le ks \qquad k = 1,2,3,\ldots$$

$$f_1(0) = f(0)\,.$$

This function dominates f. We now go one step further and define f_2 as

$$f_2(x) = e^{-\frac{\sqrt{2}}{\sigma_2}(x-s)} f(0) \qquad x \ge 0\,.$$

This expression can be written as

$$f_2(x) = c_1 e^{-c_2 x} \qquad x \geq 0\,. \tag{B.7}$$

where c_1 and c_2 are positive constants. It is this function, which dominates both f_1 and f , that we use to prove the integrals in Eqs. (B.1) and (B.2) converge.

Since both f_P and g_P are continuous they attain a maximum in any closed and bounded set[7]. Hence both $\int_{\frac{s}{2}} g^2(x) f(x)\,dx$ and $\int_{\frac{s}{2}} g(x) f(x)\,dx$ are finite. From Eq. (B.3) it is clear that if we can show $var\ g_{Gau_{\sigma_1}}$ and $E\ g_{Gau_{\sigma_1}}$ are finite then the proof is complete. This is implied by showing that $\int_P^\infty x f_2(x)\,dx$ and $\int_P^\infty x^2 f_2(x)\,dx$ converge. But

$$\int_P^\infty c_1 x e^{-c_2 x}\,dx = \frac{c_1}{c_2} e^{-c_2 P} \left(P + \frac{1}{c_2}\right) < \infty$$

and

$$\int_P^\infty c_1 x^2 e^{-c_2 x}\,dx = \frac{c_1}{c_2} e^{-c_2 P} \left(P^2 + \frac{2P}{c_2} + \frac{2}{c_2^2}\right) < \infty$$

and the theorem is proved.

REFERENCES

[1] A. Traganitis, "Narrowband Filtering, Estimation and Detection of Non-Gaussian Processes," *Ph.D. Dissertation,* Dept. of Electrical Engineering and Computer Science, Princeton University, Princeton, N.J., 1974.

[2] H. Vincent Poor and Chein-I Chang, "A Reduced-complexity Structure For the Detection of Stochastic Signals," to appear in *Journal of the Acoustical Society.*

[3] H. Vincent Poor, Chien K. Chiang and Gary L. Wise, "On the SNR Analysis of Nonlinear Detection Systems," private correspondence.

[4] S.D. Silvey, *Statistical Inference,* Chapman and Hall, London, 1975.

[5] N.L. Johnson and S. Kotz, *Distributions in Statistics: Continuous Univariate Distributions,* John Wiley and Sons, New York, 1971.

[6] S. Czarnecki, "Nearly Optimal Detection of Signals in Non-Gaussian Noise," *Ph.D Dissertation,* Dept. of Electrical Engineering and Computer Science, Princeton University, Princeton, N.J., 1983.

[7] H.L. Royden, *Real Analysis,* Macmillan Publishing Co., Inc., New York, 1968.

[8] Robert J. Marks, Gary L. Wise and Douglas G. Haldeman, "Further Results on Detection in Laplace Noise," *Proceedings of the Twentieth Midwest Symposium on Circuits and Systems,* August 15-17, 1977.

[9] M. Kanefsky and J.B. Thomas, "On Polarity Detection Schemes with Non-Gaussian Inputs," *Journal of the Franklin Institute,* Vol. 280, pp. 120-138, 1965.

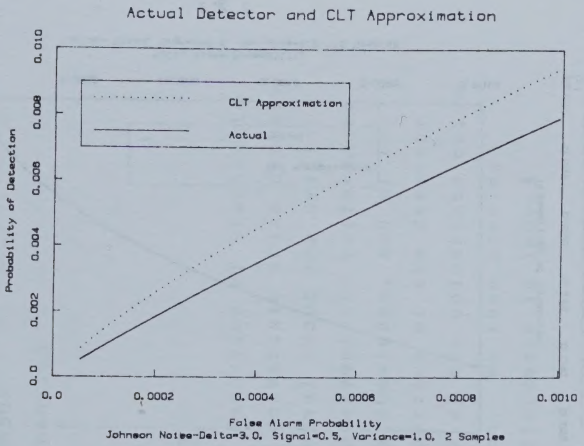


Fig. 2

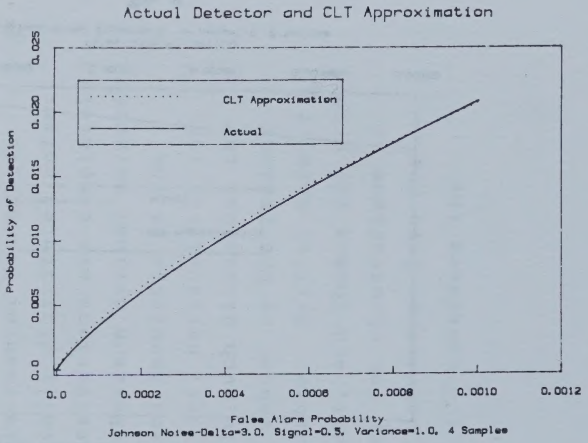


Fig. 3

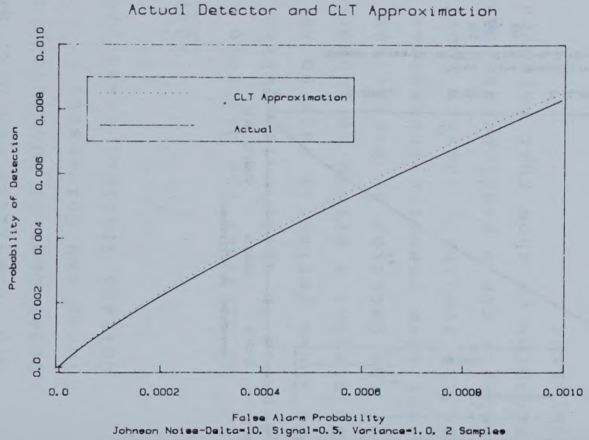


Fig. 4

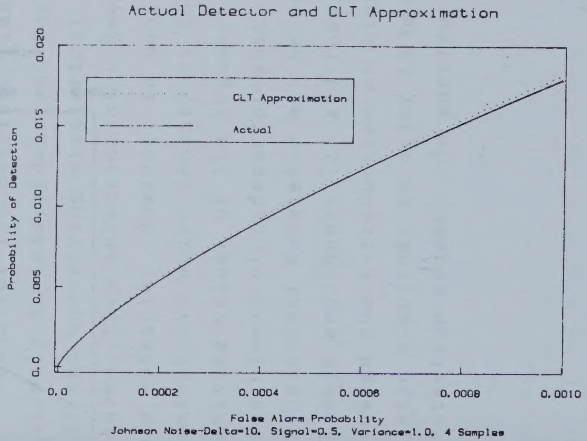


Fig. 5

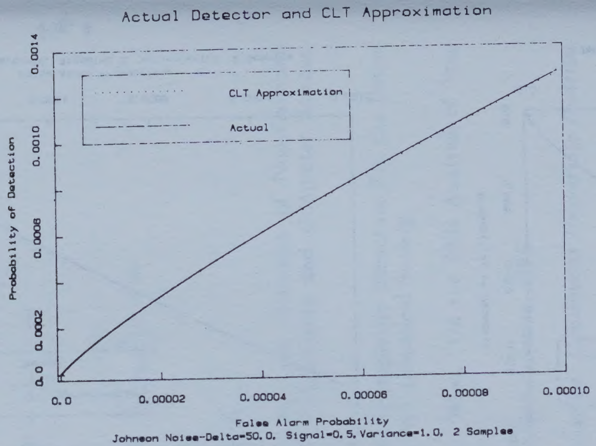


Fig. 6

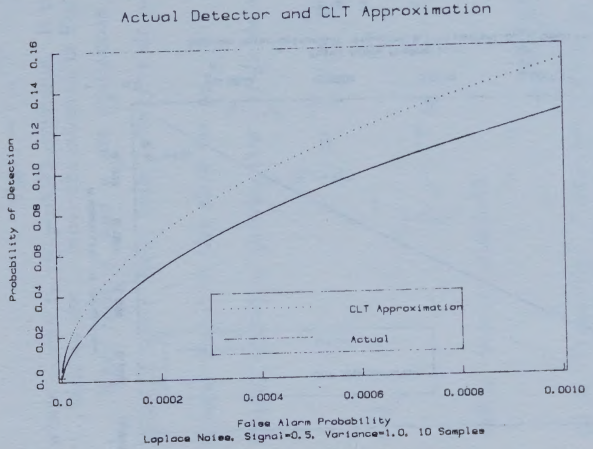


Fig. 7

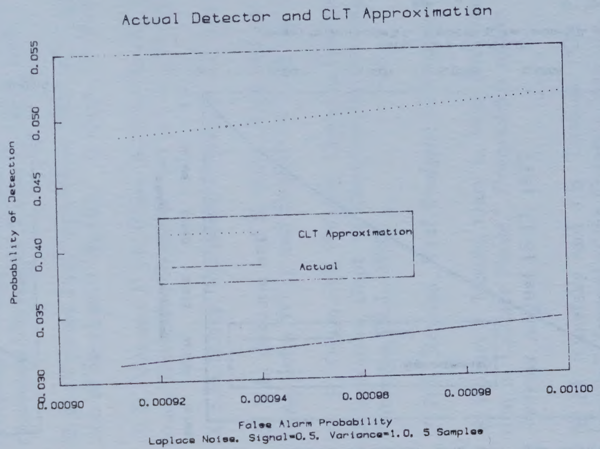


Fig. 8

ASYMPTOTICALLY ROBUST DETECTION OF STOCHASTIC SIGNALS IN CONTAMINATED NOISE

M.S. SCHNITZER and D.R. HALVERSON
Department of Electrical Engineering
Texas A&M University
College Station, Texas 77843

ABSTRACT

We consider the discrete time detection of stochastic signals in white noise, where the univariate noise density is known perfectly only on an interval about the origin. We present a method to enhance the asymptotic performance of the detector by exploiting this knowledge, and at the same time preserve robustness properties of the detector to the remaining inexact knowledge of the univariate noise density via a saddlepoint condition. We then provide examples to show that improved performance is indeed obtained.

I. INTRODUCTION

Consider the discrete time detection of stochastic signals in independent and identically distributed noise. If the underlying statistical distributions are completely known, then selection of a Neyman-Pearson optimal detector is possible. However, in many cases knowledge of the univariate noise density is imperfect, particularly for extreme values of its argument. In such situations the employment of a detector which is robust to the inexact statistical knowledge may be desirable. Earlier work of Kassam and Thomas [1] and Poor, Mami, and Thomas [2] points toward the introduction of limiting for observations of large magnitude in order to impart robustness into the detection scheme. In particular, [2] considers the asymptotically robust detection of stochastic signals in imperfectly known noise.

In this paper we consider further applications of detectors possessing the same property of limiting. We note that often more is known about the univariate noise density near the origin than on the tails, and such knowledge has the potential to be exploited for improved performance. In some earlier work [3] we show how to design the robust detector by exploiting this knowledge for improved performance over the detector of [1]; this work is applicable to the case of a time varying deterministic signal embedded in imperfectly known noise. We now consider the analogous extension of [3] for improved performance over the detector of [2]. Our results illustrate how such improved performance can be obtained and still preserve the robustness of the detector to the remaining inexact knowledge of the univariate noise density through the employment of limiting.

II. DEVELOPMENT

Suppose we denote the i.i.d. discrete time noise process by $\{N_i; i=1,2, \ldots\}$, with $f(\cdot)$ denoting the associated univariate noise density, and let the stochastic signal be denoted by $\{S_i; i=1,2,\ldots\}$; the zero mean independent signal process is assumed to be independent of the noise. The detection problem then reduces to a choice between

$$H_0 : Y_i = N_i \; ; \; i = 1, 2, \ldots, n$$

$$H_1 : Y_i = N_i + \theta S_i \; ; \; i = 1, 2, \ldots, n$$

where we observe realizations y_i of the random variables Y_i. As in [1-3], we assume the noise density $f(\cdot)$ is a member of the Huber-Tukey mixture class, i.e. $f(\cdot)$ can be expressed as $f(.)=(1-\varepsilon)f_0(.)+\varepsilon \cdot h(.)$, where ε is a fixed known number between zero and unity, $f_0(\cdot)$ is a known symmetric nominal density, and $h(\cdot)$ is an unknown "contaminating" density. In addition, however, we assume that $f(x)$ is known exactly for each x on an interval [-a, a], i.e. $h(x) = f_0(x)$ for all x in [-a, a]. We note in passing that a similar assumption is employed in [2] without being directly exploited. Motivated by the results of [1-3], as well as the form of the locally optimal detector given by [2], we employ the class of detectors where a test statistic of form

$T_g(y) = \sum_{i=1}^{n} E\{S_i^2\} g(y_i)$ is compared to a threshold, with the nonlinearity

$$g(\cdot) \text{ given by } g(x) = \begin{cases} f_0''(x)/f_0(x) & \text{if } |x| \le k \\ f_0''(k)/f_0(k) & \text{otherwise} \end{cases}$$

We thus see that specifying k effectively specifies the detector.

In order to analyze and compare the performance of the detector, and hence determine k for best performance, an asymptotic fidelity criterion is employed. Because of its relationship to the Pitman-Noether theorem [4], the criterion of Asymptotic Relative Efficiency (ARE) is particularly useful; the associated efficacy functional η provides a convenient measure of performance, where η is defined by

$$\eta(k,h) \triangleq \frac{c\left[\int_{-\infty}^{\infty} g''(x)\, f(x)\, dx\right]^2}{\int_{-\infty}^{\infty} g^2(x)\, f(x)\, dx}, \text{ with}$$

$$c = \lim_{n \to \infty} \frac{\sum_{i=1}^{n}\left[E\left\{S_i^{\,2}\right\}\right]^2}{n} .$$

As in other work which employs the ARE, the results of this paper are valid only when various mild regularity conditions are imposed on the underlying statistical distributions. These assumptions are imposed to guarantee that our expressions exist and, when necessary, are nonzero; additionally, regularity assumptions are required to insure the relevancy of the efficacy functional to the ARE via satisfaction of the hypothesis of the Pitman-Noether theorem [4]. In this paper, our results are presented under the tacit assumption of these numerous though obvious mild conditions. We refer the interested reader to [5-7], in which analogous assumptions are explicitly listed. It should be noted that we do not impose symmetry assumptions on the contaminating density $h(\cdot)$.

To preserve the robustness property of the detector, k is then chosen according to $\sup_k \inf_h \eta(k,h)$, where dependence of η on k and $h(\cdot)$ is indicated. Such a choice of k will be called an optimal choice. The following result provides insight into how to obtain an optimal choice of k:

Theorem: Let

$K = \{ k>0:$ if $k>a$ then $|g(\cdot)|$ and $|g'(\cdot)|$ are weakly monotone$\uparrow$ on $[a,k] \}$. Then a necessary and sufficient condition for k to be an optimal choice over K is that k yield the supremum among

$$\text{(a)}\quad \inf_h \eta(k,h) = \frac{2c\left[\int_0^a g''(x)f_0(x)\,dx + (1-\varepsilon)\int_a^k g''(x)f_0(x)\,dx\right]^2}{\left[\int_0^a (g^2(x)-g^2(k))f_0(x)\,dx + (1-\varepsilon)\int_a^k (g^2(x)-g^2(k))f_0(x)\,dx + \tfrac{1}{2}g^2(k)\right]}$$

if $k>a$

$$\text{(b)}\quad \inf_h \eta(k,h) = \frac{2c\left[\int_0^k g''(x)f_0(x)\,dx\right]^2}{\int_0^k (g^2(x)-g^2(k))f_0(x)\,dx + \tfrac{1}{2}g^2(k)}$$

if $k \le a$.

Proof: Consider first the proof for part (a). Note that by negative scaling when necessary we can assume without loss of generality that $g'(\cdot)$ is weakly increasing on $[a,k]$, i.e. $g''(x) \geq 0$ on $[a,k]$. Moreover, by symmetry we also have $g''(x) \geq 0$ on $[-k,-a]$. In a manner similar to the proof of the Theorem of [3] we obtain by direct evaluation

$$\eta(k,h) = \frac{c\left[2\left[\int_0^a g''(x)f_0(x)dx + (1-\varepsilon)\int_a^k g''(x)f_0(x)dx\right] + \varepsilon\left[\int_a^k g''(x)h(x)dx + \int_{-k}^{-a} g''(x)h(x)dx\right]\right]^2}{\left[2\int_0^a (g^2(x)-g^2(k))f_0(x)dx + 2(1-\varepsilon)\int_a^k (g^2(x)-g^2(k))f_0(x)dx + g^2(k) + \varepsilon\left[\int_a^k (g^2(x)-g^2(k))h(x)dx + \int_{-k}^{-a}(g^2(x)-g^2(k))h(x)dx\right]\right]}.$$

Noting the weak monotonicity of $|g(\cdot)|$ implies that $g^2(\cdot)$ is weakly increasing, we observe that the numerator of the above is minimized while the denominator is simultaneously maximized if $h(x)=0$ on $[-k,-a] \cup [a,k]$. The expression given for $\inf_h \eta(k,h)$ in part (a) then follows directly.

For part (b) a similar evaluation when $k \leq a$ yields

$$\eta(k,h) = \frac{c\left[2\int_0^k g''(x)f_0(x)dx\right]^2}{\left[2\int_0^k g^2(x)f_0(x)dx + 2(1-\varepsilon)g^2(k)\int_k^\infty f_0(x)dx + \varepsilon g^2(k)\left[\int_k^\infty h(x)dx + \int_{-\infty}^{-k} h(x)dx\right]\right]}.$$

Noting that $\int_k^{\infty} h(x)dx + \int_{-\infty}^{-k} h(x)dx = 1-2\int_0^k f_0(x)dx$,

we obtain an expression for $\eta(k,h)$ which is independent of $h(\cdot)$ and which agrees with the expression given for $\inf_h \eta(k,h)$ in part (b). QED

We remark that differentiation of the expressions of the Theorem could be employed as in [3] for the purpose of obtaining a condition on an optimal choice k as a solution of a nonlinear equation. However, we have found that for this stochastic signal situation the appropriate equations are typically so complicated that it is no more difficult to search directly over the $\inf_h \eta(k,h)$ values. We also remark that the techniques of [3] could be employed to extend the Theorem to the case of an asymmetric nominal density; the chief cost for such an extension would be the considerably more cumbersome expressions for the $\inf_h \eta(k,h)$ quantity.

We now consider an example which employs a zero mean unit variance sech noise nominal density, i.e. $f_0(x) = \frac{1}{2}\operatorname{sech}(\frac{1}{2}\pi x)$ for all x.

We compare the "worst case" efficacy $\eta_{2,w}$ of the detector of [2] (i.e. the minimal efficacy over those $h(\cdot)$ satisfying our constraints as well as those of [2]) with the "worst case" efficacy $\eta_{1,w}$ of our detector with k chosen via the Theorem. The results, which depend on the choice of a and ε, are tabulated below:

Table: Values of $\eta_{1,w}/\eta_{2,w}$

	ε=0.1	ε=0.3	ε=0.5	ε=0.7
a=2.0	6.02	2.65	1.44	1.13
a=1.5	6.02	2.65	1.44	1.13
a=1.0	3.45	2.73	1.44	1.13
a=0.5	2.73	1.55	1.15	1.04

Note that the most dramatic improvement in performance occurs when the greatest certainty is present concerning the univariate noise density (small ε and large a). This is probably consistent with what we might expect, however it is also directly opposite to what was observed for the Gaussian example of [3] for the time varying deterministic signal case. Other such opposing effects are known to exist when comparing the stochastic and deterministic signal cases, e.g. respective symmetry and skew symmetry of the nonlinearity g(·), and thus the results illustrated here need not be viewed as surprising when compared to those of [3].

III. CONCLUSION

We have introduced an approach toward improving the asymptotic performance of a detector of a stochastic signal by exploiting available knowledge of the univariate noise density near the origin, while at the same time retaining the robustness of the detector to inexact knowledge of the tails of the noise density. We have shown by way of example that such improved "worst case" performance does indeed occur.

ACKNOWLEDGEMENT

This research was supported by the Air Force Office of Scientific Research under Grant AFOSR-82-0033.

REFERENCES

1. S.A. Kassam and J.B. Thomas, "Asymptotically robust detection of a known signal in contaminated non-Gaussian noise," IEEE Trans. Inform. Theory, vol. IT-22, pp. 22-26, Jan. 1976.

2. H.V. Poor, M. Mami, and J.B. Thomas, "On robust detection of discrete time stochastic signals," J. Franklin Inst., vol. 309, pp. 29-53, Jan. 1980.

3. D.R. Halverson, "Asymptotically robust detection in partially contaminated noise," Circuits, Systems, and Signal Processing, vol. 3, pp. 493-503, 1984.

4. G.E. Noether, "On a theorem of Pitman," Ann. Math.Stat., vol. 26, pp. 64-68, 1955.

5. H.V. Poor and J.B. Thomas, "Memoryless discrete-time detection of a constant signal in m-dependent noise," IEEE Trans. Inform. Theory, vol. IT-25, pp. 54-61, Jan. 1979.

6. D.R. Halverson and G. L. Wise, "Discrete-time detection in ϕ-mixing noise", IEEE Trans. Inform. Theory, vol. IT-26, pp. 189-198, March 1980.

7. D.R. Halverson and G. L. Wise, "Asymptotic memoryless detection of random signals in dependent noise", J. Franklin Inst., vol. 312, pp. 13-29, July 1981.

A LEAST FAVORABLE DEPENDENCY STRUCTURE FOR LARGE SIGNAL DETECTION PERFORMANCE

John S. Sadowsky
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

ABSTRACT

It is desired to determine the worst possible error probability performance of a given detector which operates in an environment with uncertain data dependency. We consider a class of Markov data processes which satisfy a one shift dependency bound and agree with a specified univariate distribution. Large deviations theory is applied to determine the least favorable process in the class, using the asymptotic exponential rate of decrease of error probability as the performance criterion. The results here agree well with previous results obtained using a more tractable performance criterion; ARE (asymptotic relative efficiency). In contrast to ARE, however, the new results presented here are closely related to finite sample error probabilities, via the finite sample Chernoff bounds.

I. INTRODUCTION

Large deviations theory, and the associated Chernoff bounds, have long been accepted as an accurate means of error probability computation for communications and detection systems. Large sample detection performance with fixed signal amplitudes is typically consistent, meaning that error probabilities vanish as the sample size becomes large. For certain consistent detection problems, it is known that when optimized the Chernoff bounds are exponentially tight. This means that the bounds vanish at the same exponential rate as the true error probability. Gaussian approximation via central limit theory, on the other hand, is not exponentially tight. In fact, the exponential vanishing rate of Gaussian approximations usually exceeds the true rate. This can lead to error probability approximations which are over optimistic by orders of magnitude. Thus, in situations of large sample sizes and resultingly small error probabilities, Chernoff bounds in many instances provide an absolutely essential means of performance analysis. This is particularly true when it is desired to operate the detector with error probabilities which are small when compared to the $\propto n^{-1/2}$ accuracy of central limit theory (the Berry-Esseén theorem).

In this paper, we introduce a method of small error probability performance analysis for dependent data. The approach here parallels the work in [1] where a nominally iid dependency contamination class of admissible data distributions is considered. We seek the process distribution in the class with the worst asymptotic

Chernoff bound, thus yielding an asymptotically *least favorable Chernoff bound* for the entire class. Dependent data large deviations theory is a somewhat less tractable performance criterion than the ARE criterion used in small signal design [1]. In order to obtain these results here, we have restricted attention to only the Markov processes. This restriction allows the application of recent developments in Markovian large deviations theory [2,4,5]. The development of least favorable Chernoff bounds under the Markovian restriction is significant. First, the result does agree well with the small signal ARE results of [1] in which considered a similar class of not necessarily Markov processes. The least favorable processes in both cases are the discrete time jump processes. Second, multidimensional modeling techniques may be applied to allow Markov processes of higher order. The least favorable Chernoff bound for our class is obtained from a nonlinear integral equation. Computation of this bound is well within capabilities of modern numerical techniques. Thus this new result promises to be a valuable practical tool for communication and detection system performance evaluation.

II. PRELIMINARIES AND STATEMENT OF RESULTS

Process Distributions

Let $(\mathbf{S},\mathcal{S})$ denote a measurable state space. A data process $\{Y_k\}$, taking values from $\mathbf{S}$, is considered. Suppose that the univariate distribution of Y_k is known, and denoted by F. F is a probability measure on $\mathcal{S}$. Now, for each fixed $0 \leq \rho < 1$, a large class of admissible process distributions is generated, and denoted by $\mathcal{C}_{F,\rho}$. Each process distribution is Markov and agrees with the specified univariate distribution F. We also impose the one shift dependency bound

$$\sup_{f} \frac{|\mathrm{Cov}[f(Y_k),f(Y_{k+1})]|}{\mathrm{Var}[f(Y_k)]} \leq \rho \tag{1}$$

where supremum above is over all Borel measurable real valued functions f such that $E[f(Y)^2] < \infty$. Hereafter, Y shall always denote an $\mathbf{S}$ valued random variable with distribution F. This is simply a notational device which allows integrations against the distribution F to be expressed as Y expectations. The covariance operator is simply $\mathrm{Cov}[X_1,X_2] \triangleq E[X_1X_2] - E[X_1]E[X_2]$ and the variance is $\mathrm{Var}[X] = \mathrm{Cov}[X,X]$.

Notice that if we put $\rho = 0$, then $\mathcal{C}_{F,0}$ consists of only the iid (independent and identically distributed) process with univariate distribution F. For this reason, we think of $\mathcal{C}_{F,\rho}$ as a *dependency contamination class* about the nominal iid process distribution.

It is convenient to specify the Markov processes in $\mathcal{C}_{F,\rho}$ by their transition operators $\pi f(y) \triangleq E[f(Y_{k+1}) | Y_k = y]$. The natural domain of these operators is the space of all complex valued functions on $\mathbf{S}$ which are absolutely integrable with respect to F. Let $\mathbf{L}^\infty(F)$ be the class of all complex valued Borel measurable functions $f: \mathbf{S} \to \mathbf{R}$ such that $\|f\| \triangleq \mathrm{ess\,sup}_{y \in R} |f(y)| < \infty$, where ess means essential or almost sure with respect to the distribution F. Since $\mathrm{ess\,sup}_y |E[f(Y_{k+1})| Y_k = y]$

$\leq$ ess $\sup_y E[|f(Y_{k+1})| \; | Y_k = y]$, we see that each operator in $C_{F,\rho}$ also maps $\mathbf{L}^\infty(F)$ into $\mathbf{L}^\infty(F)$. We shall hereafter take $\mathbf{L}^\infty(F)$ to be the domain of the transition operators, with $\|\cdot\|$ denoting the essential supremum norm, or the associated strong operator norm.

Chernoff Bounds and Large Deviations Theory

Consider the error probability performance problem for detectors with memoryless test statistics of the form

$$T_n = \frac{1}{n} \sum_{k=1}^{n} g(Y_k) . \tag{2}$$

The function g is to be a bounded real valued Borel measurable function on **S**. A decision is made in favor of a given statistical hypothesis if the observation of T_n falls in the associated decision region of **R**. The complements of decision regions shall be referred to as *error sets.* If E_i is the error set associated with a statistical hypothesis H_i, then the probability of error under H_i is just $P_i(T_n \in E_i)$ where P_i is a probability distribution associated with H_i. Typically, threshold tests are employed for binary decisions. In this case, the two error sets are semi-infinite intervals. As statistical tests may be randomized on the decision region boundaries, one can redefine error set to be the closure of the decision region complement. In this case, $P_i(T_n \in E_i)$ is actually an error probability upper bound, as the correct decision may randomly be chosen on the boundary.

Hereafter, we shall consider error probability performance for a fixed hypothesis, hence we shall drop the hypothesis index i in the notation.

The Chernoff error probability bound for an error set $E \subset \mathbf{R}$ is

$$P(T_n \in E) \leq \exp(- \inf_{t \in E} I_n(t)) . \tag{3}$$

The Chernoff exponent function $I_n(t)$ is given by

$$I_n(t) \triangleq \sup_{s \in \mathbf{R}} \{ st - \mu_n(s) \} \tag{4}$$

where

$$\mu_n(s) \triangleq \log(E[\exp(s T_n)]) \tag{5}$$

is the log of the moment generating function of T_n. The right side of definition (4) is called the *convex conjugate,* or sometimes the *Legendre/Fenchel transform* [2]. This is often written $I_n(t) \triangleq \mu_n^*(t)$ where the superscript $*$ to denote convex conjugate. Since the nonlinearity g is assumed to be bounded, $\mu_n(s)$ exists and is analytic on all of **R**. Furthermore, by differentiation, $\mu_n(s)$ is convex on **R**. Thus, as is well known, the supremum in (4) is obtained at the value s which is the unique solution of $\mu_n'(s) = t$. $I_n(t)$ is also convex, and it is minimized at $E[T_n] = E[g(Y)]$. So if the point $t = E[g(Y)] \notin E$ (which is required for consistent detection) then $I_n(t)$ is minimized on the boundary of E. For example, if E is a semi-infinite interval corresponding to a threshold test on T_n, then by convexity $I_n(t)$ is minimized at the threshold. We also note, for iid data it is well known that $\mu_n(s) = n\,\mu_1(s)$ and $I_n(t) = n(st - \mu_1(s))$ where s solves $\mu_1'(s) = t$.

Recently, the large deviations for Markov processes has been considered by several authors in the mathematical literature [2,4,5]. At the core of these studies is an expression for $\mu_n(s)$ in terms of the transition operator π. This expression is

$$\mu_n(s) \quad = \quad \log(\ E[\ (e^{sg}\pi)^n\ 1(Y_1)\ ]\ ) \tag{6}$$

where $e^{sg}\pi$ is the operator (on $\mathbf{L}^\infty(F)$) which maps the function f(y) into exp(s g(y)) π f(y). Formula (6) is obtained directly from successive conditioning and the Markov property (see, for example, Lemma 5.7 in [5, p. 105]). The operator $(e^{sg}\pi)^n$ operating on the constant function 1, yields a function in $\mathbf{L}^\infty(F)$. The expectation in (6) is over the distribution of Y_1. Thus (6) holds for possibly transient situations which occurs when the initial distribution of Y_1 is not F, as well as for the case of stationary data where Y_1 has distribution F.

In cases of practical interest, the Chernoff bound (3) and the true error probability vanish exponentially in n. The key theorems of asymptotic large deviations theory essentially state that when optimized, the Chernoff bound is *exponentially tight*. This means that the exponential rate of decrease of the probability and the bound are the same. The asymptotic theory is based on the function

$$\mu(s) \ \triangleq \ \overline{\lim_{n\to\infty}} \ \frac{\mu_n(s)}{n} \tag{7}$$

and its convex conjugate

$$I(t) \ \triangleq \ \mu^*(t) \ = \ \sup_{s\in\mathbf{R}} \ \{\ st - \mu(s)\ \}. \tag{8}$$

I(t) is called the large deviations *rate function,* and from Theorem 1 below it will be clear that I(t) gives the exponential rate of decrease of error probabilities. As a limit of convex functions, $\mu(s)$ is also convex. In (7), the limit is defined to be $+\infty$ in the case that $\mu_n(s)/n$ diverges to $+\infty$. Define $\mathcal{D}(\mu) \triangleq \{s\colon \mu(s) < \infty\}$. We say $\mu(s)$ is *steep* if it is differentiable on $\mathcal{D}(\mu)$ and $|\mu'(s)|$ approaches ∞ as s approaches the boundary of $\mathcal{D}(\mu)$. We now state a general large deviations theorem, in a form sufficient for our application. This theorem is taken from a recent paper by Ellis [2].

Theorem 1: Given $\mu(s)$ as defined in (7), suppose that $\mathcal{D}(\mu)$ has a nonempty interior containing the point s = 0. Then for all closed sets E,

$$\overline{\lim_{n\to\infty}} \ \frac{1}{n} \log(\ P(T_n \in E)\ ) \ \leq \ - \inf_{t\in E} I(t). \tag{9a}$$

Furthermore, if definition (7) exists as a limit for all s $\in \mathcal{D}(\mu)$, and if $\mu(s)$ is differentiable and steep, then for all open sets E

$$\underline{\lim_{n\to\infty}} \ \frac{1}{n} \log(\ P(T_n \in E)\ ) \ \geq \ - \inf_{t\in E} I(t). \tag{9b}$$

The infimum of I(t) in (9a) and (9b) is called the *Cramér transform* associated with the set E [2]. If both (9a) and (9b) hold and E has no isolated points, then applying (9a) to the closure of E and (9b) to the interior of E we obtain the same upper and lower bound. This follows because I(t) is nonnegative and convex, hence continuous on its effective domain $\{t: I(t) < \infty\}$. Note also that in this case the effect of test randomization on decision region boundaries is asymptotically negligible in this case. Equality (9a) and (9b) is the precise statement of *exponential tightness* for the Chernoff bound (3).

Results

From (8) we see that an upper bound for $\mu(s)$ will yield a lower bound for I(t). Thus given that the first conditions of Theorem 1 holds for a lower bound of I(t), then the upper bound (9a) will hold with I(t) replaced with its lower bound. Recall that for a continuous operator A on $\mathbf{L}^\infty(F)$, $||A||$ denotes the strong operator norm associated with the essential supremum norm. From (6), we have an upper bound for the Markov case; $\mu_n(s) \le E[\, ||(e^{sg}\pi)^n||\ ||1||\,] = ||(e^{sg}\pi)^n||$. Applying this bound to (7) we obtain

$$\mu(s) \;\le\; \lim_{n\to\infty} \log(\, ||(e^{sg}\pi)^n||^{1/n}\,) \;=\; \log(\, r(e^{sg}\pi)\,) \tag{10}$$

where $r(\cdot)$ denotes the spectral radius of the argument operator. The existence of a limit and the equality in (10) is a consequence of the spectral radius formula, Theorem 10.12 in [6, p. 235].

Theorem 2 below gives an extended spectral radius bound for the entire dependency contamination class $C_{F,\rho}$. By the arguments above, this yields an upper bound (9a) for the entire class. Let $\tilde{\pi}_\rho$ be the transition operator for the discrete time Markov jump process. This is a process such that Y_{k+1} is equal to Y_k with probability ρ, otherwise Y_{k+1} begins a new run with an independent sample from the distribution F. Specifically, $\tilde{\pi}_\rho$ is defined by mapping

$$\tilde{\pi}_\rho\, f(y) \;=\; \rho\, f(y) + (1-\rho)E[f(Y)] \tag{11}$$

for all $f \in \mathbf{L}^\infty(F)$. Note that $\tilde{\pi}_\rho \in C_{F,\rho}$. In fact, $\tilde{\pi}_\rho$ obtains the dependency bound (1) for all real $f \in \mathbf{L}^\infty(F)$.

Theorem 2: Let v be a bounded Borel measurable real valued function on **S**, and let $\bar{v}$ denote the ess sup v with respect to the distribution F. Define

$$f_\lambda(y) \;\triangleq\; (\,\lambda \exp(-v(y)) - \rho\,)^{-1}\,. \tag{12}$$

If

$$E[\,(\,\exp(\bar{v} - v(Y)) - 1\,)^{-1}\,] \;>\; \frac{\rho}{1-\rho}\,, \tag{13}$$

then for all $\pi \in C_{F,\rho}$,

$$r(\,e^v\pi\,) \;\le\; r(\,e^v\tilde{\pi}_\rho\,) \;=\; \lambda$$

where $\lambda > \rho \exp(\bar{v})$ is the unique solution of

$$E[f_\lambda(Y)] \;=\; \frac{1}{1-\rho}\;. \tag{14}$$

Condition (13) is actually quite mild. Since the expectation is strictly positive, (13) holds for all bounded v with sufficiently small ρ. It is always satisfied if $P(v(Y) = \bar{v}) > 0$, because in this case the expectation is ∞.

Putting $v = sg$, Theorem 2 gives an asymptotic Chernoff bound (9a), via the spectral radius bound (10), which holds for the entire class $C_{F,\rho}$. Notice that with $\rho = 0$, (13) is always satisfied and (14) reduces to $\lambda = E[\exp(v(Y))]$, or $\lambda(s) = E[\exp(sg(Y))]$. This agrees with the iid large deviations theory. Our next result will demonstrate that the Chernoff bound for $\tilde{\pi}_\rho$ is exponentially tight ((9a) meets (9b)), and also agrees with the spectral radius bound of Theorem 2. Thus, since $\tilde{\pi}_\rho \in C_{F,\rho}$, we shall say that the Chernoff bound for $\tilde{\pi}_\rho$ is *asymptotically least favorable* and *tight* for the class $C_{F,\rho}$. To demonstrate tightness, we use the classification of $e^v\tilde{\pi}_\rho$ as an R-positive recurrent non-negative kernel which is given in Theorem 3 below. This classification is quite technical. We rely heavily on the general theory non-negative operators. The classification and a key result (Corollary 6.4) is taken from a recent book by Nummelin [3]. Due to the technical nature and length, we omit the proof of Theorem 3.

Theorem 3: Suppose that condition (13) holds for a given bounded and Borel function v, and let $\lambda > \rho \exp(\bar{v})$ be the unique solution of (14). Then the operator $e^v\tilde{\pi}_\rho$ is R-positive recurrent with convergence parameter $R = 1/\lambda$. Furthermore, the following limit exists

$$\lim_{n\to\infty} \frac{\log(\,E[\,(e^v\tilde{\pi}_\rho)^n\, 1(Y_1)\,]\,)}{n} \;=\; \log(\,\lambda\,). \tag{15}$$

We conclude this section by applying Theorems 1, 2 and 3 to the case of memoryless detection with a test statistic generated by a bounded Borel measurable function g as in (2). In Theorems 2 and 3 we consider $v = sg$ for arbitrary $s \in \mathbf{R}$.

Theorem 4: Consider a test statistic as given by (2) where $g\colon \mathbf{S} \to \mathbf{R}$ is bounded and Borel measurable. In addition, suppose that

$$P(\,g(Y) = \bar{g}\,) > 0 \quad \text{and} \quad P(\,g(Y) = \underline{g}\,) > 0 \tag{16}$$

where $\bar{g}$ = ess sup g and $\underline{g}$ = ess inf g with respect to the distribution F. Then with $v = sg$, for all $s \in \mathbf{R}$ equation (14) has a unique solution $\lambda(s) > \rho \exp(\max(s\bar{g}, s\underline{g}))$, the function $\mu(s) = \log(\,\lambda(s)\,)$ has domain $\mathcal{D}(\mu) = \mathbf{R}$ and is differentiable with derivative

$$\mu'(s) \;=\; \frac{E[\,g(Y)\exp(-sg(Y))\, f_{\lambda(s)}(Y)^2\,]}{E[\,\exp(-sg(Y))\, f_{\lambda(s)}(Y)^2\,]}\;. \tag{17}$$

Furthermore, with $I(t) = \mu^*(t) = st - \mu(s)$ where s is the unique solution of $\mu'(s) = t$, the upper bound (9a) holds for all processes in $C_{F,\rho}$ and the lower bound (9b) holds for the jump process $\tilde{\pi}_\rho \in C_{F,\rho}$.

In Theorem 4, condition (16) is a sufficient, but not necessary, condition for (13) to hold for all $s \in \mathbf{R}$. Condition (16) is satisfied by generally satisfied by robust detectors, as some degree of limiting is usually required for robustness.

III. AN EXAMPLE: THE SIGN DETECTOR

In this example, we suppose that $g(y) = \pm 1$, with $P(\, g(Y) = 1\,) = p$ and $P(\, g(Y) = -1\,) = q = 1 - p$. If $\mathbf{S} = \mathbf{R}$, this situation occurs with the sign detector; $g(y) = \mathrm{sgn}(y)$ (we suppose that $P(Y = 0) = 0$). Equation (14) is reduced to the quadratic

$$\lambda(s)^2 - b(s)\lambda(s) + \rho \quad = \quad 0.$$

where

$$b(s) \quad \triangleq \quad [(1-\rho)p + \rho] \exp(s) \; + \; [(1-\rho)q + \rho] \exp(-s)\,. \tag{18}$$

The unique solution $\lambda(s) > \rho \exp(|s|)$ is clearly

$$\lambda(s) \quad = \quad \frac{1}{2}\,[\; b(s) \; + \; \sqrt{b(s)^2 - 4\rho}\;]\,. \tag{19}$$

Now suppose that the statistical decision is to be performed by comparing the statistic T_n in (2) against a threshold. We consider only the case in which the threshold is 0, which is the usual case in a spread spectrum communications receiver for example. Since the Chernoff bound is optimized at $t = 0$, we need to compute $I(0)$. This is equivalent to solving $\lambda'(s) = 0$, which in turn implies that we must solve $b'(s) = 0$. The solution is

$$s \quad = \quad \log\left(\sqrt{\frac{(1-\rho)q + \rho}{(1-\rho)p + \rho}} \right).$$

Substituting the above solution into (18), we obtain

$$b(s) \quad = \quad 2\sqrt{[(1-\rho)p + \rho][(1-\rho)q + \rho]} \quad = \quad 2\sqrt{(1-\rho)^2pq + \rho}$$

because $p + q = 1$. Thus from (8) and (19) we have

$$I(0) \quad = \quad -\log(\; \sqrt{(1-\rho)^2pq + \rho} \; + \; (1-\rho)\sqrt{pq}\;). \tag{20}$$

It can be checked that $I(0) = 0$ in the case $p = q = 1/2$, as it must (the first term in the log reduces to $(1 + \rho)/2$). Also, equation (20) clearly reduces to the well known result $I(0) = -\log(\,2\sqrt{pq}\,)$ for the iid case $\rho = 0$.

V. CONCLUSION

The results presented here introduce a new approach to performance analysis of large sample - small error probability detection systems in the presence of spurious data dependency. The formulation of our least favorable dependency approach closely parallels the previous work in [1] where the performance performance criterion was Pitman's ARE. Both here and in [1] it was determined that the least favorable dependency structure is the jump process with run length distribution determined by the dependency parameters. The performance criterion here, the large deviations asymptotic rate function, is much less analytically tractable than ARE. As a result, while the formulation and conclusions here parallels [1], the proofs and analytical techniques used here are completely different. Comparing the two works, we first note that the present work demonstrates, at least in the context of Markov processes, that conclusions obtained using the much more tractable ARE performance criterion do extend to a large signal setting. Conversely, the broader scope obtained in [1] (no Markovian condition) suggest that the present large signal result may be extended to remove the Markovian restriction. The direct approach is modeling of higher order Markov dependency as vector Markov processes. At present, we leave multidimensional Markov modeling as a topic for future research.

APPENDIX: SKETCH OF PROOF OF THEOREM 2

Due to the length, it is impossible to present a complete proof for Theorem 2 here. Instead, we briefly describe the basic steps.

Step 1: A spectral radius upper bound. For any $\pi \in \mathcal{C}_{F,\rho}$, $e^v\pi$ is a continuous operator on $\mathbf{L}^\infty(F)$. Let $\sigma(e^v\pi)$ denote its spectrum. For any $\lambda \in \sigma(e^v\pi)$ there exists an $f \in \mathbf{L}^\infty(F)$ such that $\lambda \exp(v)\, f = \pi\, f$. Using this fact, the dependency bound (1) and the triangle inequality can be applied to develop the upper bound

$$r(e^v\pi) \leq \sup \ \overline{r}(f) \tag{A1}$$

where

$$\overline{r}(f) \ \triangleq \ \frac{\rho\, E[f(Y)^2] \ + \ (1-\rho)\, E[f(Y)]^2}{E[\exp(-v(Y))\ f(Y)^2]}. \tag{A2}$$

Notice that $\overline{r}(\cdot)$ defined in (A2) depends only on ρ, F and function v. The dependence on particular transition operators enters only through the function f associated with the spectrum element λ. Having taken the supremum over f, the bound (A1) holds for all $\pi \in \mathcal{C}_{F,\rho}$. The supremum in (A1) is over all non-negative and bounded functions f. The bound (A1) still holds if we extend the domain of the supremum to all real valued functions f that $E[f(Y)^2] < \infty$ not identically 0 (almost surely F). This extension of domain allows the calculus of variations to be applied to determine the supremum. We shall see that the maximizing f is bounded and non-negative.

Step 2: Functional optimization. To compute the supremum on the right side of (A1), we first determine the form of the function which maximizes $\overline{r}(\cdot)$. Consider minimization of the denominator $E[\exp(-v(Y))f(Y)^2]$ while constraining the values of $E[f(Y)]$ and $E[f(Y)^2]$. Using the techniques of the calculus of variations, we determine that

$$f(y;\eta_1,\eta_2) \triangleq \frac{\eta_1}{\exp(-v(y)) - \eta_2} \tag{A3}$$

solves this constrained optimization. The parameters η_1 and η_2 are Lagrange multipliers. Furthermore, this function yields a constrained minimum provided the condition

$$\eta_2 < \exp(-\bar{v}), \tag{A4}$$

holds. Under this condition, the Hamiltonian is strictly convex.

Step 3: Parametric maximization. Now, to maximize $\bar{r}(\cdot)$ we need only maximize $\bar{r}(\, f(\,\cdot\,;\eta_1,\eta_2)\,)$ with respect to η_2. Note that $\bar{r}(\cdot)$ is independent of η_1. It is convenient to select η_1 as a function of η_2 so that $E[f(Y;\eta_1,\eta_2)] = 1$. Introducing the parameter $\eta_3 = E[\, f(Y;\eta_1,\eta_2)]$ one can obtain

$$\xi = \xi(\eta_1,\eta_2,\eta_3) \triangleq \bar{r}(\, f(\cdot;\eta_1,\eta_2)\,) = \frac{\rho\eta_3 + (1-\rho)}{\eta_1 + \eta_2\eta_3}.$$

It is then demonstrated that the total derivative with respect to η_2 (using the fact that η_1 and η_3 are functions of η_2) vanishes at a point η_2^*. Furthermore, this total derivative is ≥ 0 for all $\eta_2 < \eta_2^*$ and ≤ 0 for all $\eta_2 > \eta_2^*$. This value η_2^* is the solution of

$$E\left[\frac{\eta_2}{\exp(-v(Y)) - \eta_2}\right] = \frac{\rho}{1-\rho}. \tag{A5}$$

If the solution of (A5) satisfies (A4), then the supremum on the right side of (A1) is $\bar{r}(\, f(\,\cdot\,;\eta_1,\eta_2^*)\,)$.

Step 4: Identification of $r(\, e^v\tilde{\pi}_\rho\,)$. At this point it is convenient to change of variables. Put $\eta_2 = \rho/\lambda$ with $\lambda > \rho\exp(\bar{v})$. With this change of variable, (A5) becomes equation (14) of Theorem 2. For each $y \in S$, $f_\lambda(y)$ is a strictly positive, continuous, strictly decreasing function of λ, and (by dominated convergence) $\lim_{\lambda\to\infty} E[f_\lambda(Y)] = 0$. Thus if condition (13) holds, then (14) has a unique solution satisfying (A4). Consider the jump process operator $\tilde{\pi}_\rho$ acting on f_λ. By (14)

$$\tilde{\pi}_\rho f_\lambda(y) = \rho f_\lambda(y) + (1-\rho)E[f_\lambda(Y)] = \rho f_\lambda(y) + 1$$

$$= \frac{\rho + \lambda\exp(-v(y)) - \rho}{\lambda\exp(-v(y)) - \rho} = \lambda\exp(-v(y))\, f_\lambda(y).$$

Thus $\lambda \in \sigma(e^v\tilde{\pi}_\rho)$ and f_λ is the associated function. Now $E[\, f(Y)\,\tilde{\pi}_\rho\, f(Y)\,] = \rho E[f(Y)^2] + (1-\rho)E[f(Y)]^2$ (for every f). Thus multiplying the above expression by f_λ and taking the expectation yields

$$E[\exp(-v(Y))f_\lambda(Y)^2] = \frac{1}{\lambda}(\ \rho E[f_\lambda(Y)^2] + (1-\rho)E[f_\lambda(Y)]^2\ ) .$$

Applying definition (A2), we then have $\overline{r}(f_\lambda) = \lambda$.

For all $\pi \in C_{F,\rho}$, we have

$$r(\ e^v\pi\ ) \leq \lambda \leq r(\ e^v\tilde{\pi}_\rho\ ) \leq \overline{r}(f_\lambda) = \lambda .$$

REFERENCES

[1] J. S. Sadowsky, "A maximum variance model for robust detection and estimation with dependent data," scheduled to appear in *IEEE Trans. Inform. Theory,* vol. IT-32, 1986.

[2] R. S. Ellis, "Large deviations for a general class of random vectors," *Ann. Prob.,* vol. 12, pp. 1-12, 1984.

[3] E. Nummelin, *General Irreducible Markov Chains and Non-negative Operators.* New York: Cambridge University Press, 1984.

[4] M. D. Donsker and S. R. S. Varadhan, "Asymptotic evaluation of certain Markov process expectations for large time, I," *Comm. Pure & Applied Math.,* vol. 28, pp. 1-47, 1975.

[5] D. W. Stroock, *An Introduction to the Theory of Large Deviations.* New York: Springer-Verlag, 1984.

[6] W. Rudin, *Functional Analysis.* New York: McGraw-Hill, 1973.

ON THE STOCHASTIC SCHEDULING OF A TREE*

Christos H. Papadimitriou† and John N. Tsitsiklis‡

ABSTRACT: *We consider the problem of optimal scheduling of a set of jobs obeying in-tree precedence constraints, when a number M of processors is available. It is assumed that the service times of different jobs are independent identically distributed exponential random variables. We show that policies of the "Highest-Level-First" type are optimal asymptotically, as the number of jobs tends to infinity.*

1. INTRODUCTION.

Scheduling jobs with equal execution times and tree-like precedence contraints to minimize makespan (latest finishing time) is a classical problem in scheduling [2]. It is known that, for any number of processors, the most natural policy, which assigns to the next available processor the leaf that has the largest height (distance from the root) is optimal [3]. This policy is called *highest-level first.*

An interesting twist on this theme is to allow the execution times of the tasks to be, not fixed and equal, but independent and identically distributed random variables, with exponential distribution [5]. The results here are less complete: It is known that the highest-level first policy is optimal *only in the two-machine case* [5]. In fact, for two machines this policy is optimal under a wide variety of criteria [1]. Unfortunately, for three machines the highest-level first policy is not optimal, and no tractable way to approach this problem is known. It was shown in [4] that a generalization of the two-machine problem in another direction (and-or precedences) is PSPACE-complete.

In this paper we show that for any number of machines the highest-level first policy, although suboptimal, is not much worse than the optimum. In particular, we show that, for any tree, the relative error of the highest-level first policy is $O(N^{-\delta})$, where N is the number of nodes in the tree and δ a positive real.

2. PROBLEM DEFINITION.

We are given a set of M processors and an in-tree with N nodes. Each node represents a job which may be executed (serviced) by any of the processors. The service time is assumed to be an exponentially distributed random variable with mean $1/\lambda$; we also assume that the service times of different jobs have the same mean and are independent. At any time some subset of the original set of jobs may have been already serviced and have been deleted from the tree. A job may be serviced only by assigning a processor to it. Moreover, processors may be assigned, at any given time, only to those jobs which correspond to the leaves of the as yet unserviced portion of the tree (these will be called free jobs). Equivalently, the branches in the original tree may be thought of

---

* Research supported by an NSF grant and an IBM faculty development award.

† Stanford University U.S.A., and National Technical University of Athens, Greece.

‡ Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

as precedence constraints on the order at which different jobs may be serviced.

A "scheduling policy" is a rule which given any in-tree selects at most M of its (free) nodes and assigns processors to them. We assume that we may reassign any processor at those times that the service of some job has been completed and, therefore, the tree is modified. On the other hand, for simplicity, we do not allow more general preemptive policies in which processors might be reassigned even if no change in the graph has occured. For the cost function under consideration (expected makespan), this assumption makes no difference, due to the exponential distribution assumption on the service times of the jobs.

Our objective is to study policies which are (nearly) optimal, that is policies which (for any given tree) minimize the expected time until the last node (the root) is serviced.

A special (and very easily implementable) class of policies is the class of "Highest-Level-First" policies. With such policies, the processors are assigned to jobs at the highest possible level. More precisely, we define the "depth" of a job to be the distance of the corresponding node from the root of the tree; with a highest-level-first policy, if a processor is assigned to a job at depth d, then any free job at depth $d+1$ or larger must have been also assigned to a processor. Notice that there exist more than one highest-level-first policies; a particular one could be chosen by following a lexicographic tie-breaking rule, for example.

It is known that highest-level-first policies are not optimal for $M > 2$. However, we will show that they come within an arbitrarily small percentage of the optimal, as the size of the original tree grows to infinity.

We start with a more convenient formulation of the problem. In the new formulation we are given the same in-tree. However, we let time be discrete. At each time n we assign the M processors to some of the free jobs. (If there are less than M free jobs left, some processors will have to remain unassigned.) Then, a processor is chosen randomly, each one having the same probability $1/M$ to be selected. If the chosen processor had been assigned to a job, then the corresponding node is deleted from the tree. The process terminates when all nodes have been deleted from the tree. Notice that, under any fixed policy, the expected termination time for the original continuous time problem is exactly equal to the expected termination time for the newly defined discrete time problem, times the factor $1/(\lambda M)$. (This is because the time until the first service completion is exponentially distributed with rate λM and this time is independent from which processor was the successfull one.) For this reason, we will restrict our discussion from now on to the above discrete time model.

Let $G = G(0)$ denote a given tree with $|G| = N$ nodes. Let π be any scheduling policy and let H be some highest-level-first policy. Let $T_\pi(G)$ and $T_H(G)$, be the termination time when policy π, H, respectively, is used. Our main result follows.

Theorem: $E[T_H(G)] \leq \min_\pi E[T_\pi(G)](1+\alpha(N))$,
where $\alpha(N)$ depends on G only through N and has the property that $\lim_{N\to\infty} \alpha(N) = 0$.

With our proof, $\alpha(N) = O(N^{-\beta})$, for some $\beta > 0$. Nevertheless, we are not able to construct any examples showing that a bound of this form is tight and we are tempted to conjecture that $E[T_H(G)] - \min_\pi E[T_\pi(G)] \leq A$, for all G, and for some $A > 0$, independent of G.

3. OUTLINE OF THE PROOF.

In this section we outline the main ideas of the proof of our result. The details may be found in [6]. The general plan of the proof is the following: we will do a sample path by sample path comparison of an arbitrary policy with a highest-level-first policy and show that the highest-level-first policy cannot be much worse. However, this sample path comparison is valid only for "typical" sample paths, for which the proportions predicted by the law of large numbers approximately hold. We then use the fact that "non-typical" sample paths have negligible probability.

A sample path for the problem under consideration may be described concisely by a sequence $S = \{s_1, s_2 ...\}$ such that $s_n \in \{1, ..., M\}$, for all n. In particular, s_n determines the identity of the randomly chosen processor at time n. Notice that all possible values of s_n are equally likely.

Definition: Let $\epsilon > 0$, $\delta > 0$ and N be given. We say that the sequence S is (ϵ, δ, N)-*typical* if the following holds. For any connected subset I of $\{1, ..., N\}$ which contains at least N^δ elements, the set $\{n \in I : \ s_n = i\}$ has cardinality at least $\frac{|I|}{M} - |I|^{\epsilon+1/2}$, for every i. (Whenever the values of (ϵ, δ, N) are clear from the context, we will simply call such a sequence typical.) Using the standard estimates for the tail of the binomial distribution we have:

Lemma 1: Given some $\epsilon > 0$, $\delta > 0$, the limit of the probability that S is not (ϵ, δ, N)-typical, multiplied by N, goes to zero, as N goes to infinity.

From now on we let $\epsilon > 0$ and $\delta > 0$ be fixed and such that $\epsilon < 1/4$, $\delta < 1/2$. We will occasionally use the notation $f = O(g)$ which stands for $f \leq cg$, for some constant c which may depend on ϵ, δ and M but not on G or S.

For a simple class of in-trees the approximate performance of highest-level-first policies is straightforward to establish:

Lemma 2: For any in-tree G of depth D and which has, for any d, at most M leaves of depth d, the termination time $T_H(G, S)$ corresponding to a highest-level-first policy H and a typical sequence S is bounded by $MD + O(D^{\frac{1}{2}+\epsilon})$. Moreover, for such trees we also have $E[T_H(G)] = MD + O(D^\alpha)$, where $\alpha < 1$.

Our next step is to extend Lemma 2 to more general trees. Let us assume that S, π, H have been fixed. We need some more notation. Let $G(n; S)$, $G'(n; S)$ be the remaining tree at time n when policy H (respectively, π) is followed and given that sequence S has occured. Let $d_1(n), ..., d_M(n)$ be the depths of the jobs assigned to the M processors at time M, under policy H. We assume that under policy H, lower numbered processors are assigned to jobs of larger depth; that is, if $i \leq j$, then $d_i(n) \geq d_j(n)$. This assumption leads to no loss of generality, given that the processors are identical. Notice that with this assumption the depth of $G(n)$ is equal to $d_1(n)$. Let $D(n)$ denote the depth of $G'(n)$. We assume that if at time n, $G'(n)$ has a single node at depth $D(n)$ and if π assigns a processor to that node then it is processor 1 which is assigned to that node.

Let $n*$ be the first time that $G(n)$ has less than M leaves. Notice that $|G(n)| = |G(0)| - n \leq |G'(n)|$, for all $n \leq n*$.

Lemma 3 below is very esay to show. Lemma 4 that follows has a more complicated proof and forms the heart of our argument.

Lemma 3: $D(n) \geq d_M(n)$, for all $n \leq n*$.

Lemma 4: $d_1(n*) - D(n*) = O(N^\alpha)$, for some $\alpha < 1$.

We now complete the proof of the Theorem. Notice that

$$E[T_H(G(0))] = E[n*] + E[E[T_H(G(n*))|s_1, ..., s_{n*-1}]]$$

and

$$E[T_\pi(G(0))] = E[n*] + E[E[T_\pi(G'(n*))|s_1, ..., s_{n*-1}]].$$

Now, using the last statement in Lemma 2,

$$E[T_H(G(n*)) - T_\pi(G'(n*))|s_1, ..., s_{n*-1}] = E[T_H(G(n*)) - T_\pi(G'(n*))|G(n*), G'(n*)] = O(d_1(n*) - D(n*)).$$

If S is a typical sequence, we have $d_1(n*) - D(n*) = O(N^\alpha)$, because of Lemma 4. If S is not typical we can still bound $d_1(n*) - D(n*)$ by N. Using Lemma 1, the contribution of non-typical sequences is therefore vanishingly small, as N increases to infinity. We now notice that $E[T_\pi(G)] \geq N$ to obtain, for some constant c,

$$E[T_H(G) - T_\pi(G)] \leq cN^\alpha \leq cE[T_\pi(G)]N^{\alpha-1}$$

which concludes the proof of the theorem.

REFERENCES

[1] J. Bruno, 'On Scheduling Tasks with Exponential Service Times and In-Tree Precedence Constraints", *Acta Informatica*. to appear, 1985.

[2] E.G. Coffman, Jr. *Computer and Job Shop Scheduling Theory*, Prentice-Hall, 1976.

[3] T.C. Hu "Parallel Sequencing and Assembly-Line Problems", *OR*, 9, pp. 841-848, 1961.

[4] C.H. Papadimitriou "Games Against Nature", *Proc. 24th FOCS*, pp. 446-450, 1983.

[5] K.M. Chandy, P.F. Reynolds "Scheduling Partial-Ordered Tasks with Negative Exponentially Distributed Times", TR-104, Univ. of Texas, Austin.

[6] C.H. Papadimitriou and J.N. Tsitsiklis, "On the Stochastic Scheduling of a Tree", Technical Report LIDS-P-1478, M.I.T., Cambridge, MA, 1985.

LAYER STRUCTURE IN SERIES-PARALLEL GRAPHS

MOHAMMED M. KOUTA
SUSAN E. CONRY
Department of Electrical and Computer Engineering
Clarkson University
Potsdam, New York 13676

Abstract

A new class of series-parallel graphs is defined. In [MAYR], it has been shown that scheduling this class on an unbounded number of identical processors is NP-complete. Here we show that there exists a polynomial time algorithm for scheduling these graphs when the number of the processors is bounded. This algorithm utilizes a decomposition based on the layer structure of the graph. Moreover, if the height of the precedence graph is bounded, then the complexity of the suggested algorithm becomes linear.

Extended Abstract

Given a set of tasks or jobs $J=\{J_i, 1 \leq i \leq n\}$, a set of identical processors $P=\{P_k, 1 \leq k \leq m\}$ and a partial order β where $J_i \beta J_j$ means that J_j cannot be started until J_i is completed. The partial order is usually presented in the form of a directed acyclic graph (dag) with the set of tasks as nodes. There is a directed path from J_i to J_j if and only if $J_i \beta J_j$. The transitive arcs are usually omitted.

We assume that any job can be executed by any of the processors and all the jobs have equal execution time. We also assume that once P_i begins to execute J_k, it works without interruption until the completion of J_k $(1 \leq i \leq m, 1 \leq k \leq n)$. The scheduling of the set of tasks J on the set of identical processors P subject to β is a mapping S from J to a subset of the nonnegative integers $\{0,1,2,...,L-1\}$ defined as follows.
(i) $J_i \beta J_k$ implies that $S(J_i) < S(J_k)$ for all $J_i, J_k \in J$.
(ii) $1 \leq |S^{-1}(r)| \leq m$ for all $r \in \{0,...,L-1\}$.
L is called the length of the schedule S and the objective is to find S with minimum length. For $r \in \{0,...,L-1\}$ r is called a _time slot_. $S(J_i)=r$ means that J_i will be executed by the schedule S during slot r. $S^{-1}(r)$ is the set of tasks executed by S during r. The slot that contains m tasks is called _complete_ or _saturated_. Otherwise it is called a _vacancy_ that contains $m-|S^{-1}(r)|$ gaps. A schedule whose slots are complete except possibly the last slot is called _full_. Clearly a full schedule is optimum.

It was shown [HU] that in-forests can be scheduled in linear time for any m. Similar results could be generalized to out-forests [MAYR]. In [PAPA] another efficient algorithm is presented for scheduling dags whose incomparability graphs are chordal.

The problem for m general is NP-complete [ULLM], [LENS] even if the dags are mixed forests or level forests [MAYR]. When m is fixed, the only known efficient algorithm that works on general dags is for m=2 [COFF], [FUJU], [GABO] and [SETH]. For $m \geq 3$ with general dag the complexity is open [ULLM], [GARE]. Recently it has been shown [DOLE.84], that there exists an $O(n^{h(m-1)+1})$ algorithm to find the optimum schedule of a general dag with height h on m identical processors. Therefore for bounded values of both h on m the algorithm is polynomial in n (Recall that n is the number of tasks).

Our method of scheduling starts by partitioning the tasks in the precedence graph D into layers $\gamma_h, \gamma_{h-1}, \ldots, \gamma_0$, where h is the height of D. The set of available tasks (the tasks that have no predecessors) form the highest layer γ_h. The layer γ_i, $0 \leq i \leq h$ consists of all the tasks that become available after removing all tasks in γ_k.

The tasks in γ_i, $1 \leq i \leq h$ are partitioned into two subsets, the working tasks and the reserve tasks. The working tasks at γ_i are those tasks that have one or more direct successors at γ_{i-1}. All the other tasks at γ_i are reserve tasks.

The working tasks at a given layer γ_i are decomposed further into subsets called the working subsets. Executing all the tasks of a working subset makes at least one task available from γ_{i-1}. The number of distinct working subsets at a given layer is called the width of the layer and is denoted by w_i. The maximum over all layers of w_i is the width of the dag. It is not difficult to find a linear algorithm that partitions the tasks in a dag into working subsets. Figures 1 through 3 show a dag, its static layer structure, working and reserve tasks and subsets for each layer.

If $|\gamma_h| > m$, then it is clear that an optimal schedule should choose m tasks from γ_h to execute first. The problem of developing a polynomial algorithm can be viewed as the problem of reducing the number of choices an algorithm can make to find the optimum schedule. Let D be a series-parallel graph. Several reductions of choices can be made in computing the minimum length schedule of D on m identical processors.

The first reduction shows that an optimum can always start by executing working tasks first. A reserve task is one whose execution cannot make any additional tasks available for execution. If an optimal schedule performs some reserve task before a working task at the same layer, it is easy to see that there is another optimal schedule that does the working task first.

The second reduction is suggested by a general result concerning optimal schedules for series-parallel graphs. It has been shown [KOU] that series-parallel graphs posess what we call the disclosure property. This means that an optimal schedule S can always be found that has the following characteristic. Once S has started executing in a working subset, it continues to choose tasks in that working subset (in preference to all other tasks) until the working subset is exhausted. The reduction in the number of choices is clear. Rather than choosing a task, the algorithm only needs to choose a working subset and execute from it until there are no tasks remaining in it.

A third reduction shows that an optimum can execute the tasks from the chosen working subset in any order. This is a direct result of the structure of series-parallel graphs and the definition of working subsets. All tasks in a working subset of a series-parallel graph have the same set of successors. Thus it makes no difference which tasks within a working subset are done first.

Let S be an arbitrary feasible schedule of length L for a bounded width dag D on m processors. If D^t is the dag left undone at the beginning of slot t, we define w^t_h to be the width of the highest layer of D^t and we denote the maximum of w^t_h over t between 0 and L-1 by F(m,h). A bounded width graph D is strictly bounded with respect to m if F(m,h) = g(m), where g is an arbitrary integer function of m. A strictly bounded dag has no more than g(m) working subsets in the highest layer of D^t.

Scheduling strictly bounded series-parallel graphs has been shown to be NP-complete for m unbounded [MAYR]. However, for m fixed and by using the three reductions discussed above, we show that strictly bounded series-parallel graphs with respect to m can be scheduled in polynomial time with complexity of $O(n^{g(m)+1})$. If the height of D is bounded, then the complexity becomes of O(n). These results are a corollary of a more general result for series-parallel graphs.

Theorem

Let D be a series-parallel graph of height h and width w such that $w \leq f(m)$. An optimal schedule of D on m identical processors can be found by an algorithm of complexity $n*(F(m,h)^m(h+2)^{F(m,h)})$.

Sketch of proof

The algorithm is a dynamic programming algorithm which, at each recursive step, chooses one of the working subsets to execute. Any gaps in the last slot are filled from other working subsets at the same level or from the highest reserve tasks. In the worst case, there are $F(m,h)^m$ possible choices for saturating the slot containing the last tasks from the chosen working subset.

It is not hard to show that the number of recursive steps is bounded by $(h+2)^{F(m,h)}$. At each step, the algorithm has no more than $F(m,h)^m$ possible choices and must recompute (in linear time) the layer structure of the remaining subdag. Thus the complexity of the algorithm is bounded by $n*F(m,h)^m*(h+2)^{F(m,h)}$.

Corollary

A series-parallel graph which is strictly bounded with respect to m can be scheduled in polynomial time on m identical processors, for m fixed.

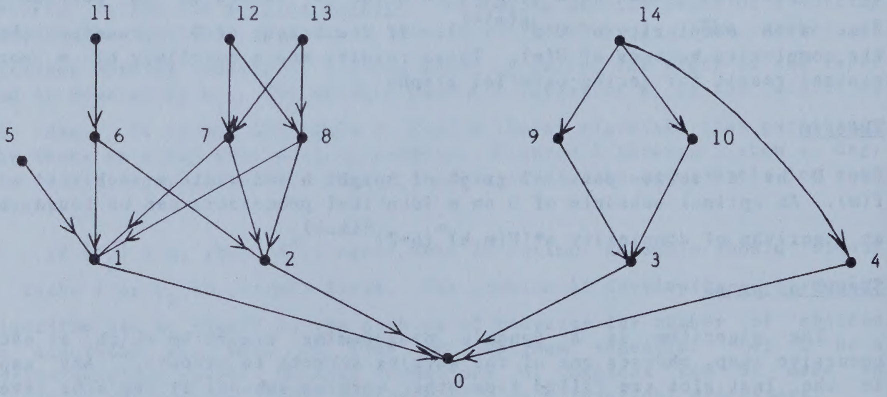
Proof

By the theorem above, since $F(m,h) \leq g(m)$ and $h \leq n$, an optimal schedule can be found using an algorithm of complexity $n*g(m)^{m}*(h+2)^{g(m)}$, which of $O(n^{g(m)+1})$.

A 'level forest' is a dag consisting of two or more components each one of which is a level graph. A dag is called a 'level graph' if its node set can be partitioned into subsets L_0, L_1, ..., L_h such that for all $1 \leq i \leq h$ there is an arc from every node in L_i to every node in L_{i-1}.

Using the above result, we show that a level forest consisting of $c < m$ components can be scheduled in polynomial time with complexity of $O(n^{m-1})$. If the height of the forest is bounded then the complexity becomes of $O(1)$.

Using the result that suggests that the problem of scheduling a dag consisting of $c \geq m$ components polynomially reduces to the problem of scheduling the highest m-1 components, we were able to show that an arbitrary level forest can be scheduled in polynomial time with complexity of $O(n^{m-1})$. Moreover, if the height of the forest is bounded then the complexity becomes of $O(n)$.



THE DIRECTED GRAPH D

$\gamma_3 = \{14, 13, 12, 11, 5\}$

$\gamma_2 = \{10, 9, 8, 7, 6, 4\}$

$\gamma 1 = \{3, 2, 1\}$

$\gamma 0 = \{0\}$

STATIC LAYER STRUCTURE OF D

FIGURE 1

$W_3 = \{14, 13, 12, 11\}$

$R_3 = \{5\}$

$W_2 = \{10, 9, 8, 7, 6\}$

$R_2 = \{4\}$

$W_1 = \{3, 2, 1\}$

$R_1 = \Phi$

$W_0 = \Phi$

$R_0 = \{0\}$

WORKING AND RESERVE TASKS AT LAYERS OF D

FIGURE 2

Working subsets of γ_3

$\{14\}$
$\{13, 12\}$
$\{11\}$

Working subsets of γ_2

$\{10, 9\}$
$\{8, 7, 6\}$

Working subsets of γ_1

$\{1, 2, 3\}$

FIGURE 3

References

[AHO] Aho, Hopcroft and Ullman. The design and analysis of computer algorithms, Addison-Wesley Publishing Company, 1974.

[BOND] Bondy and Murty. Graph theory with applications, North Holland Inc., 1976.

[BUER] Buer and Mohring. A fast algorithm for the decomposition of graphs and posets, Math of O.R., Vol. 8, no. 2, 1983, pp. 170-184.

[COFF] Coffman and Graham. Optimal scheduling of 2-processors, Acta Info. 1, 3, 1972, pp. 200-213.

[DOLE.84] D. Dolev and M.K. Warmuth. Scheduling Precedence Graphs of Bounded Height, Journal of Algorithms 5, 48-59, 1984.

[DOLE.80] D. Dolev. Scheduling Wide Graphs. Technical Report STAN-CS-80-832, Department of Computer Science, Stanford University, December 1980.

[FUJU] Fuju, Kasami and Ninomiya. Optimal sequencing of two equivalent processors, SIAM J. Appl. Math. 17, 4, 1969, p. 784.

[GABO] H. Gabow. An almost linear algorithm for two processors scheduling, JACM Vol. 29, no. 3, July 1982, pp. 766-780.

[GARE.83] M.R. Garey, D.S. Johnson, R.E. Tarjan and M. Yannakakis. Scheduling Opposing Forests, SIAM J. Algebraic Discrete Meth. 4, 72-93, 1983.

[GARE.79] Garey and Johnson. Computers and Intractability, W. H. Freeman and Company, 1979.

[GOYA] D.K. Goyal. Scheduling Series-parallel Structured Tasks on Multiprocessor Computing Systems. Technical Report CS-76-034, Computer Science Department, Washington State University, Pullman, Washington, September 1976.

[GRAH.1969] R.L. Graham. Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. Vol. 17, no. 2, March 1969, pp. 416-429.

[GRAH.1972] R.L. Graham. Bounds on multiprocessing anomalies and related packing problems, AFIPS Conf. Proc. 40, 1972, pp. 205-217.

[HANS] P. Hanson. The architecture of concurrent programs, Englewood Cliffs, 1977.

[HOPC] Hopcroft and Ullman. Introduction to automata theory, language and computation, Addison-Wesley Publishing Company, 1979.

[HU] T.C. Hu. Parallel Sequencing and assembly line problems, Oper. Res. 9, 1961, pp. 841-848.

[KOU] M.M. Kouta. Scheduling Unit Execution Time Tasks Subject to Precedence Constraints, Ph.D. Dissertation, Electrical and Computer Engineering Department, Clarkson University, Potsdam, NY, August 1985.

[LAU] H.A. Lauwerier. Asymptotic Analysis, Mathematisch Centrum, Amsterdam, 1977.

[LENS] J.K. Lenstra and H.G.R. Kan. Complexity of Scheduling Under Precedence Constraints, O.R., Vol. 26, No. 1, 1978, pp. 22-35.

[MAYR] E. Mayr. Well structured parallel programs are not easier to schedule, Department of Computer Science, Stanford University, September 15, 1981.

[PAPA] C.H. Papadimitriou and M. Yannakakis, Scheduling interval-ordered tasks, SIAM J. Comp. 8, 1979, pp. 405-409.

[SETH] R. Sethi. Scheduling graphs on Two-processors, SIAM J. Comp. 5, 1, 1976, p. 73.

[ULLM] J.D. Ullman. Np-complete scheduling problems, J. Comp. and System Science 10, 1975, pp. 384-395.

[VAL] J. Vales. Parsing Flowcharts and Series-Parallel Graphs, Technical Report STAN-CS-78-682, Computer Science Department, Stanford University, Stanford, CA, 1978.

[WARM.1980] M.K. Warmuth. M processors unit-execution time scheduling reduced to M-1 weakly connected components, Masters thesis, Department of Computer Science, University of Colorado, Boulder, December 1980.

[WARM.1981] M.K. Warmuth. Scheduling on profile of constant breadth, Ph.D. thesis, Department of Computer Science, University of Colorado, Boulder, August 1981.

ANOMALOUS BEHAVIOR IN BIN PACKING ALGORITHMS

FRANK D. MURGOLO*
Department of Mathematics and Computer Science
University of California, Riverside, CA 92521

INTRODUCTION

It has become common when working with approximation algorithms for combinatorial optimization problems to analyze the performance ratio of the algorithm under study. If we let $A(I)$ be the measure of the performance of a given algorithm for an instance I of a particular combinatorial optimization problem and $OPT(I)$ be the measure of an optimal algorithm for the same instance, then the ratio of $A(I)$ to $OPT(I)$ provides us with an indicator of the quality of the given algorithm.

It has been shown that certain algorithms, although having comparatively good performance ratios, possess the undesirable property of sometimes performing "worse" when their inputs and/or constraints are made "better". For example, in the common bin-packing problem we are given a list of items L and asked to pack them into as few bins as possible. We form a new list L' by deleting some elements of L and/or reducing the size of some elements of L. If an algorithm never uses more bins to pack L' than it uses to pack L we say that algorithm is monotonic, otherwise we say the algorithm is non-monotonic. It could be important to know that an algorithm has the characteristic of possibly performing worse when seemingly favorable changes are made to its inputs.

In [Gr66] it was shown that certain multiprocessor scheduling algorithms may behave in a anomalous manner. There it is shown that changing the task list in an apparently favorable fashion, i.e., decreasing the time required for each task or increasing the number of processors, can increase the time it takes for the task list to be completed. Following that, [Gr72] and [Jo73] showed that the bin packing heuristics First-Fit and First-Fit Decreasing are also anomalous. In [Gr66] upper and lower bounds on anomalous behavior were given, whereas in [Gr72] and [Jo73] only examples of anomalies were provided.

[FGG78] examines the paging algorithms Page Fault Frequency and Working Set providing empirical results of a case study. No analysis of the anomalies found was attempted.

In the present paper we extend this study of anomalous behavior and examine many of the popular algorithms employed for the bin packing problem. To do this we introduce a technique which allows one to completely characterize the monotonic/anomalous behavior of any algorithm in a large, natural class. Using this technique we show that Next-Fit and Next-Fit-2 are monotonic. We show that for $k \geq 3$, Next-Fit-k is non-monotonic. We show that Best-Fit, Best-Fit Decreasing, Worst-Fit and Worst-Fit Decreasing are also non-monotonic. We show that if a list L of items is such that Best-Fit, First-Fit or Worst-Fit can pack L into at most two bins then for any list L_0, where L dominates L_0, these algorithms will never use more than two bins to pack L_0. (We say L dominates L_0 if L_0 can be created by reducing the size of some items of L and/or deleting some items of L.) Our examples of the non-monotonic behavior of Best-Fit and Worst-Fit and the example of [Jo73] for First-Fit each require only three bins for the larger list L. Therefore, we see the increase from two to three bins can cause these algorithms to go from being monotonic to non-monotonic. We then examine a variation of Next-Fit-2. Instead of using

---

* Supported by National Science Foundation Grant MCS 79-04997 while the author was at the University of California, Irvine.

a First-Fit strategy when packing the last two allocated bins, as Next-Fit-2 does, we try the Best-Fit strategy. We do this to see if the restriction of examining only the last two allocated bins is sufficient for ensuring monotonic behavior. We prove that it is not. We do the same analysis with a Worst-Fit strategy and again find that this modified Next-Fit-2 is not monotonic. Therefore it is the restriction of only packing into the last two allocated bins in conjunction with the First-Fit strategy which causes Next-Fit-2 to be monotonic.

Monotonicity Results			
Algorithm	Monotonic ?	Excess Bins	Excess Bins
Name	Y-N	Lower Bound	Upper Bound
Next-Fit	Y	NA	NA
Next-Fit-2	Y	NA	NA
Next-Fit-k, k>2	N	1	n
First-Fit	N	$n/75$	n
First-Fit-Decreasing	N	1	n
Best-Fit	N	$n/42$	n
Best-Fit-Decreasing	N	1	n
Worst-Fit	N	$n/15$	n
Worst-Fit-Decreasing	N	1	n

Table 1. A summary of some of the results presented in this paper. Here n is the number of bins used by the algorithm to pack the list of items Y where $Y \ dom \ X$. The excess bins is the difference between the number of bins used to pack X and the number of bins used to pack Y.

Having found some algorithms to be non-monotonic we wish to find bounds on this type of behavior. ([Gr66] analyzes bounds for non-monotonic multiprocessor scheduling algorithms.) For any reasonable algorithm we provide upper bounds on non-monotonic behavior. (A reasonable algorithm is one which never packs an item into an empty bin if it can fit into an already allocated bin.) For some particular reasonable algorithms, First-Fit, Best-Fit and Worst-Fit, we show this bound is tight to within constant factors. Finally, we give conditions under which any reasonable algorithm will always be monotonic. Table one summarizes some of the results obtained in the paper.

References

[FGG78] Franklin, M.A., Graham, G.S. and Gupta, R.K., "Anomalies with Variable Partition Paging Algorithms" *Communications of the ACM*, **21**, 3, 1978, pp. 232-236.

[Gr66] Graham, R.L., "Bounds for certain multiprocessing anomalies," *Bell System Technical Journal*, **45**, 9, 1966 pp. 1563-1581.

[Gr72] Graham, R.L., "Bounds on multiprocessing anomalies and related packing problem," *Proceedings of the AFIPS Spring Joint Computer Conference*, May 16-18, 1972 pp. 205-217.

[Gr76] Graham, R.L., "Bounds on Performance of Scheduling Algorithms," in *Computer and Job Shop Scheduling*, E.G. Coffman, Jr. (ed.), John Wiley & Sons, Inc. New York, 1976, pp. 165-178.

[Jo73] Johnson, D.S., *Near optimal bin packing algorithms*, Doctoral Thesis, Mass. Inst. of Tech., Project MAC, Cambridge, Mass., 1973.

A FAST LINEAR SPACE ALGORITHM FOR COMPUTING LONGEST COMMON SUBSEQUENCES

A. APOSTOLICO and C. GUERRA
Department of Computer Science
Purdue University
West Lafayette, IN. 47907

ABSTRACT

Computing only the length of the longest common subsequence(s) (LCS) of two strings of lengths m and $n \geq m$, respectively, can be achieved in space $\Theta(n)$ by most of the algorithms proposed in the past. However, the only known algorithm that also computes an LCS in linear space is one by Hirschberg which never takes less than $\Theta(nm)$ time, i.e., the worst case time lower bound established for the LCS problem when the alphabet is unrestricted and the model of computation consists of the (fairly general) decision tree with $[=, \neq]$ comparisons. The algorithm proposed in this paper requires linear space to compute the LCS, and yet it can be expected to take considerably less than $\Theta(nm)$ time in most practical cases.

1. PRELIMINARIES

We consider *strings* α, β, γ, ... of *symbols* on an *alphabet* $\Sigma = \{ \sigma_1, \sigma_2, \ldots, \sigma_s \}$. A string γ is a *subsequence* of α if γ can be obtained from α by deleting a certain number of (not necessarily consecutive) symbols. Let $\alpha = a_1 a_2 \ldots a_m$ and $\beta = b_1 b_2 \ldots b_n$, with $m \leq n$. The *longest common subsequence* (LCS) *problem* for input strings α and β consists of finding a common subsequence γ of α and β of maximum length.

We denote by $L[i,j]$ $(1 \leq i \leq m, 1 \leq j \leq n)$ the length of an LCS between $\alpha_i = a_1 a_2 \ldots a_i$ and $\beta_j = b_1 b_2 \ldots b_j$. Fig. 1 below displays the L-matrix for the strings $\alpha = abcdbb$ and $\beta = cbacbaaba$. We use this figure to recall basic concepts and notation. An ordered pair of positions of α and β is a *match* iff $a_i = b_j$. The r entries that correspond to *matches* are encircled in figure 1. Each encircled number is the *rank* of the underlying match. Emboldened circles circumscribe the d *k–dominant* matches for the various values of the rank k. A match $[i,j]$ is k–dominant if it has rank k, and for any other pair $[i',j']$ of rank k, either $i' > i$ and $j' \leq j$, or $i' \leq i$ and $j' > j$. The boundaries in the figure separate regions with constant L-entry.

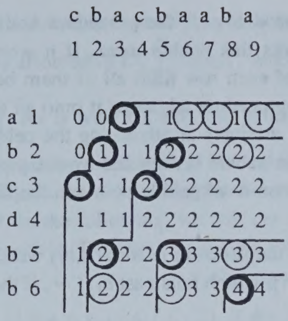


Figure 1

Let l be the length of an LCS of α and β. It is seen [HI] that computing the k-dominant matches (k=1,2,...,l) is all is needed to solve the LCS problem. In fact, once all k-dominant matches are available in suitable form, then $O(m)$ time suffices to retrieve γ. Most known approaches require $\Theta(n+d)$ space to compute an LCS, although linear space would suffice if one wanted to compute only the length l of the LCS. By contrast, the dynamic programming implementation presented in [HC] takes never more than $\Theta(n)$ space, though never less than $\Theta(nm)$ time, the worst case lower bound for the LCS problem within the unrestricted-alphabet, [=, ≠]-comparison decision tree model of computation [AH]. The remainder of this paper is organized as follows. In the next section, we present the procedure *length*, which computes the length of an LCS on any pair of substrings of α and β. This procedure is adapted from an algorithm for computing the LCS recently introduced in [AG], which has a time bound of $O(ml \cdot \min[\log s, \log m, \log(2n/m)])$. We then show that *length* and its companion procedure *lengthrev* can be cast in a recursive construct, similar to that in [HC], and based on the two recursive procedures *lcs* and *lcsrev*. The overall strategy yields an LCS using only linear space, yet it is subject to a time bound that equals that of *length* up to an additive term $O(m \log m)$. Deleting from Σ all symbols which do not occur both in α and β, does not change the solution(s) to an instance of the LCS problem. Thus we can assume henceforth without loss of generality that $s \le m$.

2. THE PROCEDURE 'LENGTH'

The procedure *length* is a direct derivation of the algorithm in [AG], except that it is devised to work on an arbitrary substrings $a_{i1},..,a_{i2}$, $b_{j1},...,b_{j2}$ of α and β, and that it does not keep a record of all dominant matches, so that it only yields the length *lsub* of the LCS for that subproblem. The procedure consists of *lsub* stages which identify the *lsub* boundaries of L in succession. It exploits the same criterion as in [HI] to trace a boundary: if $[i,j]$ is a k–dominant match then $[i',j']$ with $i'>i$ is a k–dominant match iff $j'<j$. The procedure is called by passing four parameters to it, namely, $i1,i2,j1$ and $j2$. It returns *lsub* and the the array *RANK* which contains the leftmost k-dominant match, for each k=1,..,l. It uses the following auxiliary structures:

– For each symbol of the alphabet σ, a list σ–*OCC* of all the occurrences of σ in β;

– A table *SYMB* defined as follows. $SYMB[j]=k$, if $b_j=\sigma_p$ and j is the k–*th* entry in σ_p–*OCC*. Thus the table *SYMB* enables constant time access to the entry in a σ–*OCC* list which corresponds to any given position of β.

– An array *PEBBLE* such that $PEBBLE[i]$ (i=1,..,m) contains a pointer to an entry of a_i–*OCC*. At the beginning, the procedure expects to find $PEBBLE[i]$ pointing to the the entry j of a_i–*OCC*, which corresponds to the leftmost occurrence of a_i in the interval $[j_1,..,j_2]$.

PEBBLE $[i]$ is then said to be *active*. If the procedure finds that *PEBBLE* $[i]$ falls outside the interval $[j1...j2]$, then it marks this pebble *dead*, if it were not already such. The procedure advances the active pebbles of each row until all of them become *inactive*. A pebble becomes inactive as soon as either the procedure advances it onto an entry of the associated a_i–*OCC* list which is larger than $j2$, or it attempts at advancing the pebble past the last entry of a_i–*OCC*. When the first case applies, the pebble is retracted by one position on the list: thus by the end of the execution of *length* each non-dead pebble points to the rightmost position that it can occupy in the interval $[j1...j2]$.

The algorithm uses also the function *closest* (σ, b_t) which for any given character σ returns the leftmost occurrence of σ in β which falls past b_t, (∞ , if there is no such occurrence).

```
Procedure length (i1, i2, j1, j2, RANK, lsub)
$RANK[k] = 0$, $k=1,2,...,(i2-i1)$; mark dead pebbles outside $[j1...j2]$;
$k = 0$
while there are active pebbles do (start stage $k+1$)
begin $T = j2+1$; $k = k+1$;
    for $i = i1-1+k$ to $i2$ do (advance pebbles)
    begin
        $t = T$;
        if $PEBBLE[i]$ is active and $a_i$–$OCC[PEBBLE[i]] < T$ then
        (update threshold, update leftmost $k$-dominant match )
            begin $T = a_i$–$OCC[PEBBLE[i]]$; $RANK[k] = T$ end;
        (advance pebble, or make it inactive)
        $PEBBLE[i] = SYMB[closest[a_i, t]]$;
        if $PEBBLE[i]$ is active and $a_i$–$OCC[PEBBLE[i]] > j2$ then
            begin $PEBBLE[i] = PEBBLE[i]-1$; make $PEBBLE[i]$ inactive end;
    end;
end ($lsub = k$).
```

The procedure *length* detects all k–dominant matches, as is readily checked. Unlike the algorithm presented in [AG], however, it records only the leftmost k–dominant match incurred for each k. This obtains the linear space bound.

All the elementary steps of *length*, with the exception of the executions of *closest*, take constant time. On an input of size $n+m$ the procedure handles at most m pebbles during each of the l stages. Thus the total time spent by *length* is $O(ml +$ total time required by *closest*$)$. The second term is obviously implementation dependent. One efficient implementation of *closest* is discussed in [AG]. It rests on two auxiliary structures which we now proceed to describe. First, we prepare, in time $\Theta(n)$, the table *CLOSE* $[1...n+1]$ which is subdivided into consecutive blocks of size s and defined as follows. Letting $p = j \bmod s$ $(j=1,...,n)$, *CLOSE* $[j]$ contains the leftmost position not smaller than j where σ_p occurs in $\beta\$$, with $\$$ a symbol not in Σ. *CLOSE* $[n+1]$ is set to $n+1$. Next, we assume that each σ–*OCC* list is assigned a *finger tree* [AG, BT, ME]. Roughly, a finger-tree is a balanced search tree which can be traversed in any direction. The finger is a pointer to any leaf in the tree. The main advantage conveyed to our discussion by the use of finger-trees is in that, with such trees, a search for an item which is displaced d positions (leaves) away from the current position of the finger can be carried out in $O(\log d)$ time. If the finger is updated to point to the last searched item at all times, then searching for m consecutive items in a tree which stores n keys is afforded in $O(\sum_{k=1}^{m} \log d_k)$, where the *intervals* d_k's are subject to the constraint that $\sum_{k=1}^{m} d_k \leq 2n$. As is well known, this yields the overall time bound of $O(m \log(2n/m))$. A finger-tree incarnation well fit to the manipulation of integers is the *characteristic tree* described in [AG]. However, more general versions such as originally introduced in [BT] would do just as well. In the following, we will use σ–*OCC* to refer to the list as well as to the finger tree associated with it. Thus we can always speak of the current position of the finger on a given list.

Lemma 1. For any given p and j $(1 \leq p \leq s, j1 \leq j \leq j2)$, if the finger falls d positions to the right of j, then $closest[\sigma_p,j]$ can be retrieved from $CLOSE[j]$ and from the σ_p–*OCC* list in time $O(min[logd, logs])$.

Proof. We prove our claim by giving explicitly a strategy to identify $closest[\sigma_p,j]$ from p and j. Clearly, $closest[\sigma_p,j]$ can be retrieved in constant time if $\sigma_p = b_j$. Assume now $\sigma_p \neq b_j$. First, compute $j' = (j \ div \ s) \cdot s + p$, where *div* stands for the integer division. Three cases have to be distinguished according to whether $j'=j$, $j'<j$ or $j'>j$. If $j'=j$ then $CLOSE[j] = closest[\sigma_p,j]$ by definition. Our strategy returns $CLOSE[j]$ as the answer in constant time. Next, suppose $j'<j$. If $CLOSE[j'] = j'' > j$, then clearly $j'' = closest[\sigma_p,j]$. Otherwise $closest[\sigma_p,j]$ is not smaller than $j'' = CLOSE[j']$ but not larger than $j''' = CLOSE[j'+s]$. Now $SYMB[j'']$ and $SYMB[j''']$ point to the corresponding entries in the σ_p–*OCC* list, and there can be no more than s entries in σ_p–*OCC* between these two entries. If the finger falls outside the interval delimited by these two entries, then $closest[\sigma_p,j]$ can be retrieved in $logs < logd$ steps by performing a binary search in this segment of σ_p–*OCC*. If, on the other hand, the finger falls inside the interval, then the search will take time proportional to $logd < logs$. The case $j'>j$ is handled along the same lines as the case just discussed. ☐

A claim equivalent to the above was already established in [AG]. In the present context, taking full advantage of Lemma 1 requires some simple additions to the procedure *length* which we now describe informally. The procedure resulting from imposing these new specifications on the old procedure will be called still *length*. The discussion of such upgrade leads to the statement of Theorem 1 below, which expresses the time complexity of our new *length*.

First, in order to keep track of the fingers we introduce a new global variable, namely, the array of integers $FINGER[1...m]$. This array is similar to the array *PEBBLE*, and it plays a somewhat dual role. At its inception, the procedure *length* expects to find the fingers pointing to the same locations as the pebbles. We add to this the following specifications.

- As soon as it is invoked, *length* spots the dead pebbles and kills the corresponding fingers. It then moves the non-dead fingers among $FINGER[i1]$, $FINGER[i1+1]$,..., $FINGER[i2]$: each such finger is brought onto the rightmost position in the interval $[j1...j2]$ that it can occupy on its corresponding σ–*OCC* lists. We will refer to this process as to the *finger positioning*. The positioning of each finger is clearly accomplished in $O(min[\log s, \log(j2-j1)])$ time through an application of *closest*. While the pebbles retain their identity on the σ–*OCC* lists, fingers set from different rows on the same σ–*OCC* list merge instead into one single *representative* finger. The initial position occupied by each representative finger (i.e., the largest compatible entry in the associated σ–*OCC* list) is called its *home*. The introduction of representative fingers obtains that, if, during a stage of *length*, the finger associated with some symbol is moved, then this move is implicitly imposed on all other elements of $FINGER[i_1 \ldots i_2]$ which correspond to the same symbol. We omit the details. This latter process of instituting representative fingers takes time proportional to (i_2-i_1).

- During the execution of each stage of *length*, the (representative) finger associated with each symbol in $[i_1 \ldots i_2]$ is reconsidered immediately following a *closest* query and the possible consequent update of the pebble (cfr. lines 8-10 of *length*). At that point, we simply set: $FINGER[i] = PEBBLE[i]$. Thus through each individual stage, the finger associated with each symbol moves from right to left. At the end of the stage, the finger is taken back to its home position. Each of the manipulations just described takes constant time.
- Finally, both fingers and pebbles are taken back to their initial (leftmost) position soon after the last stage of *length* has been completed. Overall, this takes time $O(i_2-i_1)$.

Theorem 1. By the combined use of *FINGER* and *CLOSE*, the procedure *length* computes the length *lsub* of an LCS of $\alpha_{i_1} \cdots \alpha_{i_2}$ and $\beta_{j_1} \ldots \beta_{j_2}$ in time $O((i_2-i_1)+lsub \cdot (i_2-i_1) \cdot min[\log s, \log(2n/(i_2-i_1))])$ and linear space.

Proof. The linear space bound is trivially achieved. As for the time, we prove a tighter bound, namely , the one obtainable from that claimed in the Theorem by replacing n with (j_2-j_1). All the operations contained in the inner loop of *length* take constant time, except for the calls to *closest* performed during the various stages, and for the i_2-i_1 initial such calls aimed at finger positionings. There can be at most i_2-i_1 calls to *closest* during each stage. We want to bound the total work performed by *closest*, on all lists and during one stage, in connection with all searches driven both by the fingers and by the table *CLOSE*. Towards this end, we can imagine all these search intervals as aligned one after the other. Since consecutive intervals on the same list overlap in at most one position, then the sum of their lengths is at most $2(j_2-j_1)$. By Lemma 1, the total work on all lists can be bounded loosely in terms of the the expression:

$$\sum_{k=1}^{i_2-i_1} \log d_k$$

where the intervals d_k are such that

$$\sum_{k=1}^{i_2-i_1} d_k \leq 2(j_2-j_1).$$

Under this sole constraint, the above sum is maximized by taking $d_k = 2(j_2-j_1)/(i_2-i_1)$. If we now take into account that, by virtue of the table *CLOSE*, it is also $d_k \leq s'$ for all values of k, then we get, for the maximum total time spent by *closest* during each of the *lsub* stages, an upper bound proportional to $(i_2-i_1) \cdot \min[\log s', \log(2(j_2-j_1)/(i_2-i_1))]$. Thus, for *lsub* stages, the bound which was claimed holds for these calls to *closest*. In finger positioning, the sum of the intervals on which representative fingers are moved is again not larger than $2(j_2-j_1)$. Thus the same argument shows that the calls to *closest* performed at the beginning for finger positioning charge a total of $O((i_2-i_1) \cdot min[\log s, \log(2(j_2-j_1)/(i_2-i_1))])$. This term is absorbed in the previous one when $lsub \neq 0$. Otherwise the procedure only spends $O((i_2-i_1))$ to find that all pebbles are dead. This concludes our proof. □

3.THE LINEAR SPACE ALGORITHM 'LCS'

Our original goal was to design an algorithm having the same time performance as *length*, yet taking linear space to retrieve also an LCS. The peculiar structure of *length* enables to adopt here profitably a divide and conquer scheme such as in [HC]. Basically, this will consist of applying the procedure *length* recursively to smaller subproblems until we obtain a trivial one.

We need to make a few additional assumptions, namely:

– We stipulate that m is a power of 2.

– We remove the previous assumption according to which, upon calling *length* with j-parameters $j1, j2$, the procedure always finds pebbles and fingers pointing to the leftmost positions in the interval $[j1...j2]$. We replace it with the new assumption that either all pebbles and fingers occupy the rightmost positions in the interval $[j1...j2]$, or else they all occupy the leftmost one. Procedure *length* checks at its inception which case applies, and brings all pebbles to their leftmost positions, if necessary. This does not affect the time bound of the procedure.

Our final algorithm is actually based on the four procedures *length*, *lengthrev*, *lcs* and *lcsrev*. The companion procedure of *length*, *lengthrev*, is simply a replica of *length* just made suitable for processing the mirror image of any subproblem on the input strings. Thus, for instance, calling *lengthrev* with parameters: $1,m,1,n$, has the same effect as letting *length* run on the reverse of the input strings. The mirror procedure *lcsrev* is related to the procedure *lcs*, which is still to be described, in the same way. In conclusion, we only need to list *lcs*.

```
Procedure lcs (1,m,1,n,LCS)
begin
  if $n=0$ or $m=1$ then determine LCS in constant time
  else (split the problem into subproblems)
     begin
        length (1,m/2,1,n,RANK1,$lsub_1$);
        lengthrev (m/2+1,m,1,n,RANK2,$lsub_2$);
        $j$ = findmax (RANK1,RANK2,$lsub_1$,$lsub_2$,lsub);
           (determine the length lsub of the LCS for this subproblem)
        lcs (m/2,m,1,j,LCS1);
        lcsrev (m/2+1,m,n-j,n,LCS2);
        combine the two outputs LCS1 and LCS2;
     end;
end.
```

The function *findmax* determines the value $j=RANK1[k]$ such that, if $j'=RANK2[k']$ is the smallest entry of $RANK2$ which is larger than j, then $lsub = k+k'$ is a maximum. Thus, the first time *findmax* is executed, it returns $lsub = l$, i.e., the length of any LCS of α and β. Moreover, the match $[i,j]$ with maximum $i \leq m/2$ belongs to an LCS of the two input strings.

The function *findmax* can be straightforwardly implemented in such a way as to require a number of steps proportional to $lsub_1 + lsub_2 \leq 2lsub$.

The proof of correctness for *lcs* follows from arguments similar to those developed in [HC].

Theorem 2. The procedure *lcs* finds an LCS in time $O(m \log m + ml \log(min[s, 2n/m]))$ and space $\Theta(n)$.

Proof. We consider all the executions of *length* and *lengthrev* involved at the k–th level of recursion of our strategy, at once. Such executions are relative to consecutive substrings of α of uniform length $m/(2^k)$, and consecutive substrings of β. Starting from the upper-left corner of the

L-matrix, each such substring of β is paired up twice with a substring of α, as sketched in Fig. 2.

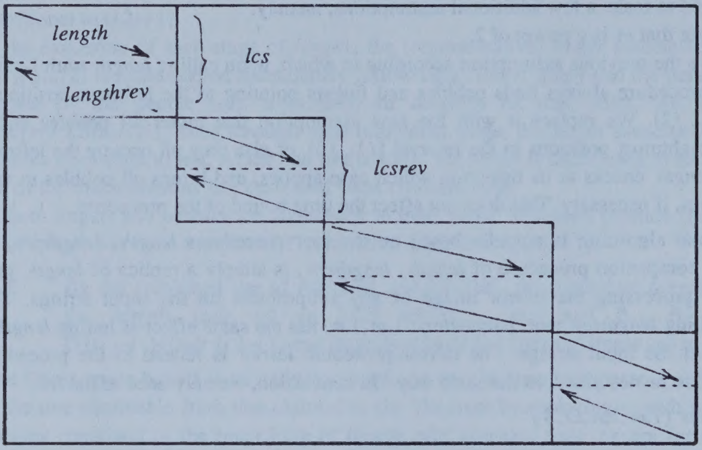


Figure 2
A possible partition of an L-matrix into rectangular subdomains whose associated subproblems have to be solved at the same depth of recursion.

The upper pairing involves an execution of *length*, the second one an execution of *lengthrev*. We define a *block* at level k as the submatrix which is the domain of two such consecutive subproblems.

All the executions of *findmax* at this level charge $O(l)$ time. Adding up for all values 1,2,...,log$m$ of k yields a bound $O(l \cdot \log m)$ for the total work performed by *findmax*.

The execution of each *length* (*lengthrev*) can be bounded in terms of $m/(2^k)+m/(2^k)\cdot l_f \log(\min[s, 2n \cdot 2^k/m])$, where l_f denotes the length of the LCS associated with the generic subproblem. There are 2^k calls at level k, yielding a total time:

$$m+\sum_{f=1}^{2^k}\frac{m}{2^k}l_f\log(min\,[s,\frac{2n}{m}2^k]),$$

up to a multiplicative constant. Now it is

$$\sum_{f=1}^{2^k} l_f \leq 2l.$$

In fact, each l_f cannot be larger than the length of the solution to the corresponding block, and the sum of the 2^{k-1} such lengths cannot exceed l,i.e., the length of the global solution.

Thus we have, in conclusion, that the total work at this level of recursion can be bounded in terms of the quantity:

$$m+l\cdot\frac{m}{2^k}\cdot\log(min\,[s,\frac{2n}{m}2^k]) \leq m+l\cdot\frac{m}{2^k}\cdot\log(min\,[s\,2^k,\frac{2n}{m}2^k]).$$

The right term can be rewritten as:

$$m + l\cdot\frac{m}{2^k}\log(2^k\cdot\min[s,\frac{2n}{m}]) = m + l\cdot k\cdot\frac{m}{2^k} + l\cdot\frac{m}{2^k}\log(min\,[s,\frac{2n}{m}]).$$

Adding up through $k = 1,2,...,\log m$ yields:

$$mlogm + ml\sum_{k=1}^{logm}\frac{k}{2^k} + ml\ \log(min\,[s,\frac{2n}{m}])\sum_{k=1}^{logm}\frac{1}{2^k}.$$

Since:

$$\sum_{k=1}^{logm}\frac{1}{2^k} = 2 - \frac{1}{2^{\log m}} < 2,$$

and:

$$\sum_{k=1}^{logm}\frac{k}{2^k} = (\log m - 1)\frac{1}{(2m)} + \frac{1}{2},$$

then we obtain the claimed $O(m\log m + ml\log(min\,[s, 2n/m])$ time bound.

The space bound for our strategy follows from an argument similar to that developed in [HC]. □

4. CONCLUDING REMARKS

We have presented an algorithm to retrieve an *LCS* of two input strings in linear space. The worst case time complexity of the algorithm is $\Theta(nm)$. However, our approach is considerably faster in favorable situations , notably, when one or more of the parameters s, m, and l is small compared to the others and to n. The only previous linear space algorithm [HC] known to these authors takes instead time $\Theta(nm)$ in all situations. Our approach exploits search techniques with the use of fingers in two ways, namely, to set up a time efficient procedure for determining the length of the solution to an arbitrary subproblem of the problem, and to randomize, up to an amortizedly negligible logarithmic cost, the initial accesses to the rectangular domains relative to the various subproblems.

Finally, we mention that new nonlinear space algorithms for the LCS problem, which also make use of some of the techniques discussed in this paper, have been introduced quite recently [AG,HD]. In particular, a strategy based on a schedule of primitive operations similar to that in [HS], but with a time bound $O(mlogn + dlog(2nm/d))$ is discussed in [AG]. Interestingly enough, it does not seem that this latter strategy is amenable to implementations that guarantee to occupy always linear space at run time.

REFERENCES

[AG] Apostolico, A. and C. Guerra. The longest common subsequence problem revisited, Tech. rep., Purdue Univ. CS Dept., (1985, submitted for publication).

[AH] Aho, A.D., D.S. Hirschberg and J.D. Ullman. Bounds on the complexity of the maximal common subsequence problem, *JACM* 23, 1, 1-12 (1976).

[BT] Brown, M.R., and R.E. Tarjan. A representation of linear lists with movable fingers. *Proceedings of the* 10*–th STOC*, San Diego, Ca., 19-29 (1978).

[HD] Hsu, W.J., and M.W. Du. New algorithms for the LCS problem, *JCSS* 29, 133-152 (1984).

[HI] Hirschberg, D.S. Algorithms for the longest common subsequence problem, *JACM* 24, 4, 664-675 (1977).

[HC] Hirschberg, D.S. A linear space algorithm for computing maximal common subsequences, *CACM* 18, 6, 341-343 (1975).

[HS] Hunt, J.W., and T.G. Szymanski. A fast algorithm for computing longest common subsequences, CACM 20, 5, 350-353 (1977).

[ME] Mehlhorn, K. Data Structures and Algorithms 1: Sorting and Searching, Springer-Verlag, EATCS Monographs on TCS (1984).

A VERY EFFICIENT ALGORITHM FOR FINDING ALL CYCLES IN A DIGRAPH

JIN,XIAOLONG

Department of Telecommunication and Computer Engineering,
Shanghai Institute of Railway Technology,Shanghai,PRC

ABSTRACT

A very efficient algorithm for finding all cycles in a digraph is developed here. The algorithm is based on the explicit use of the properties in depth-first search on a digraph.During a depth-first search to find strongly connected components,each time a chordis scanned the algorithm find all cycles led by this chord.

Having found some properties and developed some efficient criteria, we develop a regular search method to reduce the failure search.The worst case complexity is better than the present best Johnson's algorithm. It is between $O(|V|+|E|)(C+1)$ and the lower bound $O(|V|+|E|+L)$, where C is the number of cycles and L is the total length of these cycles.

1. INTRODUCTION

Finding all cycles in a digraph is a fundamental problem for some applications such as the study of flow through computer program,code optimization feedback control system and electrical networks.

Several algorithms have been proposed for this problem[1],[4], among them the best is Johnson's[2]. His algorithm is based on a careful pruning of the search of all cycles to ensure that every cycle is generated exactly once and litter backtracking is paid. The complexity of this algorithm is $O(|V|+|E|)(C+1)$, where C is the number of found cycles.

surely the lower bound for finding all cycles in a digraph will be $O(|V|+|E|+L)$, where L is the total length of found cycles and the length of a cycle is counted by the number of edges in this cycle.

We use the terminology and notations of graph theory in Deo's book[3].

Due to the definition of strongly connected component,there is no cycle among several strongly conncted components. So we have the fact

Fact 1 For a digraph G,the set of all cycles is the union of the sets of all cycles in each strongly conncted component of G.

Depth-first search(DFS) is one of the most nature way to traverse all vertices and edges systematically. With depth-first search,many basic problems such as connectivity,biconnectivity,strongly connectivity and planarity test etc, have got optimal algorithms[1].

We use an example(Figure 1) to show that it is efficient to find all cycles during a depth-first search to determine the strongly connected components of G, which has been discussed in many books. The main idea is to label the vertices of G during DFS.

(1) DFN(v)--- the visited order during the DFS.

(2) LOW(v)--- a function of vertex v which is defined as
$\mathrm{LOW}(v)=\mathrm{MIN}\{\mathrm{DFN}(v),\mathrm{DFN}(w)\mid$ w is the smallest numbered vertex in the same strongly connected component as v that can be reached zero or more tree edges followed by at most one back edge or cross edge $\}$

We assume the digraph G=(V,E) be represented by adjacency linked lists The depth-first search starts from an arbitrary chosen vertex. When the DFS reaches vertex v, an edge (v,w) belongs to one of the following four types:

(1) Tree edge--- DFN(w)=0,denoted by boldface line in Fig.1.

(2) Back edge--- $\mathrm{DFN}(v)>\mathrm{DFN}(w)$ and w is an ancestor of v.In Fig.1 (3,2), (6,1)are back edges which are denoted by lightface line

(3) Cross edge---$\mathrm{DFN}(v)>\mathrm{DFN}(w)$ but w is not an ancestor of v. In Fig.1 (4,3),(6,4),(7,6) are cross edges in crossed lines.

(4) Forward edge--- DFN(v)< DFN(w).Here (1,4),(1,7) are forward edges which are denoted in dotted lines.

When a depth-first search is going on,all the cycles can be found.In Fig.1 when vertex 3 is visited,cycle (3,2,3) is found due to the back edge (3,2).When vertex 4 is visited,cycle (4,3,2,4)is found due to the cross edge (4,3).After scanning all going out edges at vertex 2,a strongly connected component composed of vertices 2,3 and 4 is found.Similarly,in another strongly connected component three cycles are found. We notice that tree edge(1,2),forward edge (1,4)and cross edge (6,4)will belong no cycle because they are the edges between two strongly connected components.

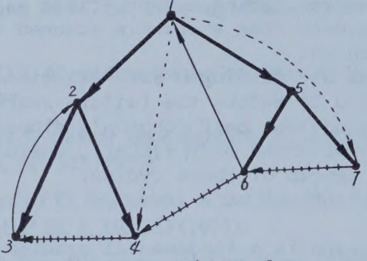


Figure 1

The paper is organized as follows: In section 2,we give the properties for efficiently finding cycles during a DFS. In section 3,we describe the algorithm and give the main procedures. In section 4, we discuss the corretness and complexity of our algorithm.

2. PROPERTIES FOR EFFICIENTLY FINDING CYCLES DURING A DFS

We will explicitly use some properties of DFS to find cycles.

First,we show the type of an edge (v,w)can be determined easyly.We use two boolean arrays INSCC and FINISH which are defined as

$$\text{FINISH}(v) = \begin{cases} \text{true} & \text{if all going out edges of v have been scanned and the DFS return to its ancestor.} \\ \text{false} & \text{otherwise.} \end{cases}$$

$$\text{INSCC}(v) = \begin{cases} \text{true} & \text{if vertes v is still in stack SCC which is employed in finding strongly connected components.} \\ \text{false} & \text{otherwise.} \end{cases}$$

Then an edge (v,w) is classified according to the criteria in Table.

type	DFN	INSCC(w)	FINISH(w)
tree edge	DFN(w)=0		
back edge	DFN(v)>DFN(w)	true	false
cross edge in a SC	DFN(v)>DFN(w)	true	true
cross edge not in a SC	DFN(v)>DFN(w)	false	true
forward edge in a SC	DFN(v)< DFN(w)	true	true
forward edge not in a SC	DFN(v)< DFN(w)	false	true

SC is the strongly connected component.

Due to fact 1, we will discuss the cycles within a strongly connected component and restrict ourself to the DFS-tree to examine its properties.

We call a sequence of tree edges $(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$, in which each v_i is the father of v_{i+1} and no vertex has two or more going out tree edges except v_1 or v_k, a tree branch. We use $v_i \Rightarrow v_j$ to denote a path from v_i to v_j. Furthermore,we use $v_i \xrightarrow{T} v_j$ to denote the path with all edges are tree edges. During a depth-first search, for any vertex v,the subtrees of v are naturely ordered from left to right according to their visited order DFN.

Lemma 1. During a depth-first search, there are cross edges from left to to right only.

Lemma 2. There is no forward edge in any leaf, especially, there are only back edges in the leftest leaf.

When a chord (v,w) is scanned during the DFS, we call the set of found cycles led by (v,w) the set of chord leading cycles. Depending on the type of (v,w), we call it the set of back, cross or forward leading cycles respectively. We characterize them with following two lemmas:

Lemma 3. In a set of chord leading cycles, any cycle C includes at least a back edge.

Proof. If the chord is a back edge, it is true.

For a cross edge or forward edge (v,w), v is the latest visited vertex and has not finished scanning. If v is the root of the DFS-tree, to form a cycle it needs an edge entering v. This is a back edge.

If v is not the root of the DFS-tree, the cycle C=(v,w,...,v) must include another unfinished vertex u, because all edges entering v have not been scanned except the tree edge (father of v,v). The edge of v which enters u is a back edge. □

Lemma 4. During the depth-first search, any chord (v,w) which is not between two strongly connected components will lead at least a cycle, where any edge in these cycles has been scanned before visiting (v,w).

Proof. We prove our lemma for each type of chord.

(1). (v,w) is a back edge. the path $w \xrightarrow{T} v$ of the DFS-tree and the back edge (v,w) form an unique cycle.

(2). (v,w) is a cross edge. suppose n is the nearest common ancestor of v and w, and l is the leaf to which there is a tree path from w. If there is a cross edge (l,w'), then we discuss w'. Otherwise, suppose (l,u) is a back edge, if u is above n then a cycle formed

$$C = (u \xRightarrow{T} n)+(n \xRightarrow{T} v)+(v \Longrightarrow w)+(w \xRightarrow{T} l)+(l,u)$$

If u is not above n, there must be a path $u \longrightarrow n$ due to the definition of strongly connected component. The cycle is

$$C = (u \Longrightarrow n)+(n \xRightarrow{T} v)+(v \Longrightarrow w)+(w \xRightarrow{T} l)+(l,u)$$

notice that all edges in C have been scanned before scanning (v,w).

(3). (v,w) is a forward edge, we go down along the DFS-tree to a leaf. With similar argument, there is at least a cycle. □

The set of back leading cycles consists of only one cycle. To find other two sets, some previously found cycles will be useful. But the storage requirement will surelyexceed what we can afford due to the fact the number of cycles in a digraph could be exponential to the number of vertices.

3.ALGORITHM

The input of the algorithm is a array of |V| adjacency linked lists ADJ of the digraph G. Our recursive procedure DICYCLE(v) performs the depth-first search. It first classifies the going out edges of the vertex v into chords and tree edges (forward edges may be inclulded, because the descendant of v has not been searched at the moment), which are linked by V(v).clink and V(v).tlink respectively. After processing all chords, we deal with tree edges. If (v,w) is a tree edge (DFN(w)=0), we recursively call DICYCLE(v). Otherwise, w is put into V(v).clink.

For a cross edge or forward edge (v,w), we start a recursive procedure PATH(v) to find the set of cross or forward leading cycles.

(1) The search of set of cross or forward leading cycles.

We start our search at w, and build a path until a vertex u which is an ancestor of v is found. Then a cycle

$$C = (v,w) + (w \Longrightarrow u) + (u \xrightarrow{T} v) \qquad (1)$$

belongs to the set of cross (or forward) leading cycles.

A stack SPATH is used to keep the so far found path. If the present path is $(w,v_1,\ldots,v_k)$, we explore all going out edges of v_k. We consider chords first. If (v_k,v_{k+1}) is one of these chords

(A) Cross edge or forward edge. We push v_k into SPATH and call PATH(v_{k+1}).

(B) Back edge. If v_{k+1} is an ancestor of v, then the cycle formed according to (1) is reported. Otherwise we recursively call PATH(v_{k+1}).

After exploring all going out chords of v_k ,we go down along one of the tree edges of v_k to traverse this tree branch. If all going out edges of v_k have been scanned,we pop the stack SPATH back to the previous vertex v_{k-1} The scanning of going out edges at v_{k-1} continues,and so on. This process continues until the stack SPATH is empty.

We get a very simple criterion to determine whether v_{k+1} is an ancestor.

Lemma 5. During the depth-first search, v_{k+1} is an ancestor of v if and only if the DFS has not finished the scanning of v_{k+1}.

Proof. At the moment v is the latest visited vertex. Due to the depth-first search nature,ancestor is laterfinished than its descendants.So all the ancestor of v have not finished its scanning, but all other vertices in depth-first searched subgraph have finished their scanning.

(2) Reduce the number of failure search.

In figure 2 we show two cases. In Fig.2(a), the back edge(w,u) is of no use for forming a cycle to any later scanned chord such as (v,w),because u is below the nearest common ancestor between v and w for later chord starting from any vertex v. But in Fig.2(b),there are several ways to form cycle For example,one cycle is

$$C = (v,w_0)+(w_0 \overset{T}{\Longrightarrow} w)+(w,u)+(u \overset{T}{\Longrightarrow} w')+(w',u')+(u' \Longrightarrow v)$$

another possible cycles could be formed through v_2 or v_1 .

We use a criterion to distinct these two cases.

```
if u has only one tree edge and its cnext=w then delete(w,u)
                                             else keep it
```

This condition corresponds to that no tree branch or chords go out in the interval [u,w). To a vertex v, cnext is a field at each its tree edge which points to the next vertex having going out chords in this tree branch This is a way to avoid exploreing vertices without going out chord.

Along the kept back edges,the search may fail. To reduce the backtrack we adopt Johnson's technique.

(a) A vertex v will be marked as unavailable by setting AVAIL(v) to false as soon as v is appended to the current path.

(b) The vertex v will remain unavailable at least until we have backed up past v to the previous vertex on the current path. If the current path up to v did not lead a cycle, v will remain unavailable for a while even after we back up past it. To prevent searching for cycles in the subgraph on which such search have be previously unsuccessful, we use B(w) to keep all predecessors of w which are unavailable vertices and not on the current path.

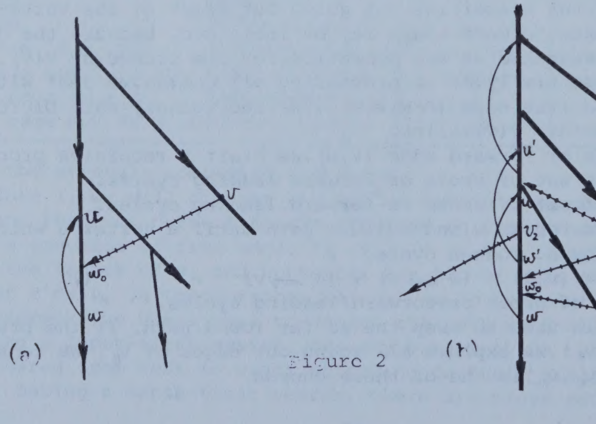


Figure 2

Using a flag=true to indicate a cycle is found, we use a recursive procedure UNMARK(u) to make u available,so do all its unavailable precedessors that are not on the current path.

(3) The main procedures

To implement above algorithm, we use following data structure:

Each vertex v is represented by a record of two fields

clink------ links to the going out chords
tlink------ links to the going out tree edges

what the tlink points to is a record with three fields:

data------- the vertex number of w of a tree edge (v,w)
cnext------ the next vertex with going out chords along this tree edge
link ------ links to next tree edge

V(1..|V|)is the array of all vertexrecords.

B(1..|V|)is the array of sets for unavailable predecessors

Two stacks:SCC and SPATH.

```
PROCEDURE  DICYCLE(v);
begin
 1   DFN(v):=num; LOW(v):=DFN(v); SCC ⇐ v; num:=num+1;
 2   for w ∈ ADJ(v) do   (* classify edges and deal with chords *)
     case    (v,w)    of
 2.1 tree edge   : put w into V(v).tlink;
 2.2 back edge   : begin  output cycle;
                      LOW(v):=LOW(w);
                      if (v has only one tree edge) and (cnext=v)
                         then delete (v,w)
                         else put w in V(v).clink
                   end;
 2.3 cross edge  : if  INSCC(w) then
                   begin  PATH(w);  LOW(v):=LOW(w);
                      for each visited vertex u do
                      begin AVAIL(u):=true; B(u) ⇐ Ø end
                   end  else  delete (v,w)
    end;
 3 if ( V(v).clink=Λ) or (|V(v).tlink|>1) then
    begin for each vertex v1 in interval (pre,v) do V(v1).tlink.cnext:=v;
         pre:=v  (* the previous vertex with chords or more than onetree*)
    end;         (* edges *)
 4 for w∈V(v).tlink do  (* deal with tree edges and forward edges *)
    if DFN(v)=0 then begin DICYCLE(w); LOW(v):=MIN{LOW(v),LOW(w)} end
      else  if INSCC(v) then
              begin  move w to V(v).clink;
                     PATH(w);
                     for each visited u  do
                     begin AVAIL(u):=true; B(u) ⇐ Ø  end
              end
              else  delete (v,w);
 5 FINISH(v):=true;
 6 if  LOW(v)=DFN(v)then pop SCC up to v include(*strongly connected *)
  end;                                         (* component reported*)
```

```
   PROCEDURE     PATH(w,flag);
   begin
1     flag:=false;  SPATH ⇐ w;  AVAIL(w):=false;
2     if V(w).clink≠∧ then
       for  u ∈ V(w).clink  do
         if AVAIL(u) then
            if  ((w,u) is back edge) and (not FINISH(u)) then
            begin output cycle according to (1); flag:=true end
            else begin PATH(u,g); flag:=flag or g end;
3     if V(w).tlink≠∧ then
        for t ∈ V(w).tlink do
        if AVAIL(t↑.cnext) then begin PATH(t↑.cnext,g); flag:=flag or g end;
4     if flag then UNMARK(w)
      else for (u∈V(w).clink)or(u∈V(W).tlink) do B(u):=B(u)∪{u} ;
5     w ⇐ SPATH
   end;

   PROCEDURE     UNMARK(u);
   begin  AVAIL(u):= true;
     for w ∈ B(u)  do
     begin  delete w from B(u);
            if not AVAIL(w) then UNMARK(w)
     end
   end;
```

The driver program to find all cycles in a digraph G= (V,E), V= 1,2...|V| using DICYCLE,PATH and UNMARK, is as follows

```
num:=1;  SPATH ⇐ Ø;  SCC ⇐ Ø;
for  i:=1 to |V| do
begin B(i) ⇐ Ø;  DFN(i):=0 end;
for i:=1 to |V| do
  if DFN(i)=0  then DICYCLE(i);
```

4.CORRECTNESS AND COMPLEXITY OF THIS ALGORITHM

Theorem 1. The algorithm correctly find all cycles of a digraph.

Proof. We prove our claim in two steps.

(1).For a chord (v,w),the procedure PATH(w) correctly find all cycles led by (v,w).

This procedure itself is in a depth-first search fasion. Starting from w, it goes to the possible farthest tree branch via cross edges. It searches each edge of a vertex on the path, so this process exhaust all possible cycles.

With the mark AVAIL, each found cycle is an elementary cycle and two found cycles differ at least in the stack SPATH or in the finally found back edges. So no duplication happens.

The detail reasoning is similar to Johnson's proof[2].

(2). The union of all sets of chord leading cycles is the set of cycle of this digraph.

We prove it by induction on the numberof chords.

During a depth-first search, the first found chord is a back edge which forms an unique cycle. This is the basis of our induction.

Suppose the theorem is true for the so far searched subgraph when the (n-1)th chord(v',w') is found.

When the n-th chord (v,w) is found, if v=v',all cycles in the set of (v, w) leading cycles differ from the previously found cycles at least at edge (v,w). If $v \neq v'$, then all edges scanned between the scanning of (v',w') and (v,w) are tree edges. If there were no chord (v,w),these edges would belong to no cycle. So the union of previously found cycles and the set of (v,w) leading cycles is the set of all cycles in the present searched subgraph. □

Now we discuss the worst case complexity of our algorithm.

Because we adopt Johnson's technique to deal with backtracking, the complexity of our algorithm will not be worse than $O(|V|+|E|)(C+1)$. But our algorithm is better at least for the following reason:

The time paid for a back leading cycle is exactly proportional to the length of back leading cycle. Let C' be the number of cross or forward leading cycles, the complexity is $O((|V|+|E|)(C'+1)+L_b)$,where L_b is the total length of back leading cycles. But there are many successful search,so it will be better.

The algorithm is implemented in PDP-11 in FORTRAN. In the test of random generate digraphs and some specialtype digraphs, it run faster than Johnson algorithm. The advantage of our algorithm over Johnson's are following:

1. Because we explicitely use the depth-first search,according to (1) a cycle is generally composed of edges in SPATH and tree edges. These tree edges need not be searched and the time paid is proportional to its length
2. We classify edges into four types, so the search is more regular and less failure would happen.
3. Because the use of cnext to link next vertex having going out chords or tree branch, the search is speeded.
4. No excessive time is paid for finding strongly connected components.

REFERENCES

[1] E.M.Reingold,J.Nievergelt,N.Deo, Combinatorial Algorithm: Theory and Practice, Prentice-hall,1977.

[2] D.B.Johnson, 'Finding all the Elementary Circuits of a Didected Graph,' SIAM J. Computing,4(1975),77-84.

[3] N.Deo, Graph Theory with Application to Engineering and Computer Science Prentice-hall ,1974.

[4] P.Mateti, N.Deo, 'On algorithms for enumerating all Circuits of a Graph' SIAM J.Computing 5(1976),90-99.

A LINEAR DIRECTED CUT TRANSVERSAL ALGORITHM FOR 1-SINK ACYCLIC GRAPHS*

K. VIDYASANKAR
Department of Computer Science, Memorial University of Newfoundland
St. John's, Newfoundland, Canada A1C 5S7

Abstract:

In a directed graph G, a directed cut transversal is a set of edges that meets every directed cut in G. A linear algorithm for finding a minimum directed cut transversal in a 1-sink acyclic graph is given.

1. TERMINOLOGY

A *directed graph* G consists of a *vertex set* VG and an *edge set* eG. Each edge is an ordered pair (u,v) of vertices, called its *ends*, and is *directed from* u *to* v. For a vertex set X, the *coboundary* δX in G is the set of edges each with one end in X, and the other in VG - X; δX is *outdirected* if for each edge (u,v) in δX, u is in X; in that case, $\delta(VG-X)$ is *indirected*. The coboundary δX is *directed* if it is either outdirected or indirected. A *directed cut* is a minimal non-null directed coboundary. Graph G is *strongly connected* if for no non-null proper subset X of VG, δX is directed.

A graph H is a *subgraph* of G if $VH \subseteq VG$, $eH \subseteq eG$, and each edge of eH has the same ends in H as in G. For a subset X of VG, G[X] is the subgraph of G whose vertex set is X and edge set consists of the edges of G whose both ends lie in X. A *strong component* of G is a maximal strongly connected subgraph of G.

A vertex x is a *source* of G if each edge of G incident to x is directed away from x; vertex x is a *sink* if each incident edge is directed toward x.

A directed graph is *acyclic* if it is free of directed cycles.

2. INTRODUCTION

A *directed cut transversal* of G is a set of edges that meets every directed cut in G.

Theorem [4]: For a finite directed graph G, a minimum directed cut transversal of G is equal in cardinality to a maximum disjoint collection of directed cuts in G.

The study of this minimax equality originated from the study of a problem posed by John Runyon [6; page 299]: for a directed graph, give an efficient algorithm for finding a minimum feedback set. A *feedback set* is a set of edges that contains at least one edge of each directed cycle of the graph. It is now known from Karp [2] that Runyon's problem is NP-complete. But when restricted to planar graphs, the problem is polynomial. This is from the fact that Lucchesi [3] has given a polynomial algorithm for finding a minimum directed but transversal in arbitrary graphs and from the following Dual Theorem which is a corollary of the above theorem.

Dual Theorem [4]: For a finite planar directed graph, a minimum feedback set has cardinality equal to that of a maximum disjoint collection of directed cycles.

The time complexity of Lucchesi's algorithm is $O(m^4 \log m)$ where m is the number of edges in the graph. More efficient methods and/or algorithms have been suggested for some special graphs [5, 7, 8] - for example, O(m) for 1-source 1-sink graphs, $O(m^2)$ for multi-source 1-sink graphs, and $O(m^3)$ for bipartite graphs. (The term 'strong cover' has also been used for directed cut transversal.) Except for the 1-source 1-sink case these algorithms do not appear to be optimal, and there are

---

* This research is supported in part by the Natural Sciences and Engineering Research Council of Canada Individual Operating Grant A-3182.

strong possibilities for improvements.

In this paper, we present an optimal algorithm, of complexity $O(m)$, for multi-source 1-sink graphs. (By directional duality, this holds for 1-source multi-sink graphs also). We restrict our attention to only acyclic graphs, since the non-trivial strong components, if any, can be found using Tarjan's depth-first search method (see [1]), and contracted to single vertices in time $O(m)$. Our algorithm is an efficient implementation of the recessional collection method of Younger [8] described below.

A *recessional* collection for 1-sink acyclic graph G is either the null set or a *successor* $\mathbb{R} \cup \{X\}$ of recessional collection $\mathbb{R}$, where X is an augment of $\mathbb{R}$. An *augment* of $\mathbb{R}$ is a minimal subset X of VG such that δX is nonnull, outdirected, and disjoint from each δY, for Y in $\mathbb{R}$. The collection $\mathbb{R}$ is *complete* if it has no successor, that is, if it has no augment.

For complete recessional collection $\mathbb{R}$, let $D = \{\delta Y : Y \text{ in } \mathbb{R}\}$. One finds a directed cut transversal t by means of an abstraction pair, defined as follows.

An *abstraction pair* $(\mathbb{R}_p, t_p)$ for complete recessional collection $\mathbb{R}$ is either $(\mathbb{R}, \phi)$ or a *successor* $(\mathbb{R}_p - \{Y\}, t_p \cup \{(u,v)\})$ of an abstraction pair $(\mathbb{R}_p, t_p)$ for $\mathbb{R}$, where Y is a set in $\mathbb{R}_p$ and (u,v) an edge of δY such that there is a directed path from y to v, where y is either the sink b of G or the end u' for some edge (u',v') in t_p. An abstraction pair is *complete* if it has no successor.

It has been shown in [8] that an abstraction pair is complete only if t_p is a directed cut transversal. Then $|t_p| = |D|$, whence t_p is a minimum directed cut transversal and D a maximum disjoint collection of directed cuts for G.

In the following, we present two subalgorithms: a labelling algorithm that outputs a complete recessional collection $\mathbb{R}$ and a maximum disjoint collection D of directed cuts, and assigns certain labels to vertices; and an abstraction algorithm that generates a minimum directed cut transversal using these labels.

3. LABELLING ALGORITHM

Algorithm L

Input: A 1-sink acyclic graph $G = (VG, eG)$ represented by arrays AT and AF of pointers to linked adjacency lists of vertices. The nodes in the linked lists have the form

VTX	LINK

where VTX is a vertex and LINK a pointer field. For each vertex w, $AT(w)$ points to the list of vertices v such that (w,v) is in eG, and $AF(w)$ points to the list of vertices v such that (v,w) is in eG. The single sink vertex is b.

Output: A complete recessional collection $\mathbb{R}$, a disjoint collection D of directed coboundaries consisting of δX for each X in $\mathbb{R}$, and nonzero *labels* for vertices of G.

Comments: The algorithm uses three stacks S, S1 and S2, all initially empty. The role of the arrays *number* and *low* is the same as in a depth-first search algorithm for finding strong components (see, for example, [1]).

Algorithm:

```
begin
  for each x in VG, number(x) ← 0 and label(x) ← 0
  i ← 1
  c ← 0
  v ← b
  number(v) ← i
  low(v) ← i
```

```
push v on S and S1
while S1 is not empty do
  begin
    while AF(v) is not nil do
      begin
        w ← VTX(AF(v))
        AF(v) ← LINK(AF(v))
        if number(w) = 0
          then begin
                 v ← w
                 i ← i+1
                 number(v) ← i
                 low(v) ← i
                 push v on S and S1
               end
          else if number(w) < number(v)
               [(v,w) is either a back edge or a cross edge]
               then low(v) ← min{low(v),number(w)}
               [else if number(w) > number(v), then
               (v,w) is a descendent edge, which is ignored]
      end
    [AF(v) is nil]
    if number(v) > low(v)
      then begin
             w ← second (from the top) vertex on S
             low(w) ← min{low(w),low(v)}
             pop v from S
             v ← w
           end
      else [number(v)=low(v); a strong component is detected]
           begin
             c ← c+1
             if number(v) ≠ 1
               then [the edges of a new directed coboundary is output]
                 begin
                   repeat
                     w ← top vertex on S1
                     while AT(w) is not nil do
                       begin
                         u ← VTX(AT(w))
                         AT(w) ← LINK(AT(w))
                         if number(u) < number(v)
                           then begin
                                  output (w,u)
                                  [directed coboundary is destroyed]
                                  enter (v,u) in AF(v)
                                end
                           [if number(u) ≥ number(v), ignore]
                       end
                       if label(w) = 0 then label(w) ← c
                       pop w from S1
                       push w on S2
                   until w = v
                   [empty the vertices of S2 into S1, and
                    output them as an augment of ℝ]
                   while S2 is not empty do
                     begin
                       x ← top vertex in S2
```

```
                          output x
                          pop S2
                          push x on S1
                        end
                    end
               else [number(v)=1; assign labels to vertices in
                    S1 and pop them]
                    begin
                      while S1 is not empty do
                        begin
                          w ← top vertex in S1
                          if label(w)=0 then label(w) ← c
                          pop S1
                        end
                      pop S
                      for each vertex x in VG output x and label(x)
                    end
                end
          end
end
```

The labelling algorithm, algorithm L, uses essentially the depth-first search (dfs) method for finding strong components in a directed graph. The dfs is started (in the reverse direction) from the single sink vertex b. The first strong component detected by the algorithm will be $G[\{v\}]$ for some source vertex v. Then $\{v\}$ is output as an augment of $\mathbf{R}$, and $\delta\{v\}$ is output as the first directed coboundary in D. Immediately, the directed coboundary is 'destroyed' as follows: for each edge (v,u) in $\delta\{v\}$, a new edge (u,v) is added to G (as an entry u in AF(v) only, and not in AT). (With the new edges $\delta\{v\}$ is not directed any longer, and hence the term 'destroyed'.) And the dfs is continued from v. Thus each time a strong component is detected, the vertex set X of the strong component is output as an augment of $\mathbf{R}$, δX is output as an element of D, δX is destroyed by adding new edges, (the procedure for destroying δX, when X has more than one vertex, is given below,) and the dfs is continued.

Let us denote the initial graph G as G_1. If G_i is the graph from which the vertex set X_i and the coboundary δX_i are output, then G_{i+1} will denote the graph obtained from G_i by adding edges that destroy δX_i. Let G_* denote G_i for some i. Then any strong component detected by the algorithm is the first one for G_*. (That is why all cross edges are relevant here.) From the properties of the dfs algorithm, when this strong component is detected, if v is the top vertex in S, then S1 contains all the vertices with nonzero number, and the vertex set X of the strong component consists of all the vertices in the top portion of S1 upto and including v. These vertices are popped from S1, output, and also stacked in S2.

It is clear that δX is outdirected, and the edges of δX are found using the adjacency lists AT(w) for each w in X. An edge (w,u) is in δX if and only if w is in X and u is not in X. If u is not in X, then either it appears below v in S1 or it is not in S1. In the latter case, number(u) = 0, and in the former, number(u) < number(v), since for any vertex x in S1, all the vertices above x in S1 have number greater than that of x, and all vertices below x have number less than that of x. Thus, in either case, number(u) < number(v), and this condition is used to output the edges of δX.

The directed coboundary δX is destroyed as follows: for each edge (w,u) in δX, a new edge (u,v) is added to the graph (again, as entry u in AF(v) only.) These edges destroy δX in the following sense. Each new edge (u,v) forms a directed cycle along with (w,u) and a directed path from v to w in the strong component. By the property that an edge is in a directed coboundary if and only if it is not in a directed cycle, neither (w,u) nor the new edge (u,v) can appear in any subsequent directed coboundaries output by the algorithm. Hence we have:

3.1 The directed coboundaries output by the algorithm are those of G, and are mutually disjoint.

After the destruction of δX, $G_\star[X]$ is no longer a strong component of $G_\star$, since $G_\star$ now includes the new edges also. And so, all the vertices X in S2 are transferred back to S1, and they appear in S1 in the same order as before. And the dfs is continued from v (which is still the top vertex in S).

It may appear that the adjacency lists AT(w) should be scanned entirely each time w occurs in X, where δX is output. Let i be the smallest index and j be another index such that both X_i and X_j are in **R** and contain w. While outputting δX_i, all the entries in AT(w) are scanned, and for each (w,u) in δX_i, (u,v) is added for the appropriate v. This implies that in G_m, for $m > i$, any strong component that meets X_i should include X_i and also all the vertices u for (w,u) in δX_i. This is true for X_j also. For the edges (w,x) not in δX_i, since w is in X_i, thus x is also in X_i and hence in X_j too. That is, no edge incident from w is in δX_j, and therefore, while outputting δX_j, AT(w) need not be scanned. For this reason, in the algorithm L, the adjacency list AT(x) for any vertex x is not scanned more than once.

In a 1-sink acyclic graph, for each vertex v, there is a directed path from v to the sink. This implies that the algorithm encounters every vertex v of the graph, and pushes it on S and S1. Hence when a strong component is detected at vertex with number 1, that is, at b, S1 must contain all the vertices of G. At this point, after proper labelling of the vertices, the algorithm is terminated.

Let $\{X_1, X_2, \ldots, X_{k-1}\}$, for $k > 1$, be the collection of vertex sets output by the algorithm. We have already seen that for each X_j, δX_j is nonnull, outdirected, and disjoint from each δX_i, for $i < j$. That X_j is a minimal such set can be verified easily. Hence each X_j is an augment. Also, G_k is strongly connected. Hence we have:

3.2 The collection $\{X_1, X_2, \ldots, X_{k-1}\}$ is a complete recessional collection.

Labelling: The algorithm assigns nonzero labels to vertices as follows: with each X_i, all the vertices of X_i that currently have label 0 are assigned label i.

It is clear that all the vertices are assigned nonzero labels. With the assumption that $\{X_1, X_2, \ldots, X_{k-1}\}$ is the recessional collection, k is the maximum label of a vertex, label of b is k, and X_k is VG. In the following, we bring out some properties concerning the labels for later use. Let D denote $\{\delta X_1, \delta X_2, \ldots, \delta X_{k-1}\}$.

3.3 For each v in X_i, $\text{label}(v) \le i$.

3.4 For each (u,v) in G, $\text{label}(u) \le \text{label}(v)$.

Proof: If label(v) is i, then v is in X_i. Since δX_i is outdirected, thus u is also in X_i. By 3.3, $\text{label}(u) \le i$.

3.5 For any edge (u,v) in G, (u,v) is in δX_i if and only if $\text{label}(u) = i$ and $\text{label}(u) < \text{label}(v)$.

Proof: Suppose (u,v) is in δX_i. Then, by the labelling method used by the algorithm, u is assigned nonzero label only after the edge (u,v) is output, the label assigned at this stage is i, and no vertex outside X_i has label i. Since v is not in X_i, $\text{label}(v) \ne i$. From 3.4 it follows that $\text{label}(v) > \text{label}(u)$.

Conversely, if $\text{label}(u) = i$, then u is in X_i; if label(v) >

label(u), then v is not in X_i, and so (u,v) is in δX_i.

3.6 In each d in D, for any two edges (u,v) and (u',v') in d, label(u) = label(u') and label(v) = label(v').

Proof: Let d be δX_i. Then, from 3.5, both label(u) and label(u') equal i. The algorithm while destroying δX_i adds edges (v,w) and (v',w) for some w in X_i. From then on, any strong component of G_* that includes w includes v and v' also. Therefore, if j is the least index such that v is in X_j, then label(v)=label(v')=j.

A modification in the algorithm L: For each vertex w, the algorithm L scans AT(w) only once, and the nonzero label to w is assigned after AT(w) has been scanned completely. Thus once a vertex receives a nonzero label, it is not required for any purpose other than for inclusion in subsequent augments of $\mathbb{R}$. Our abstraction algorithm uses only the labels of the vertices and does not need the recessional collection itself (as can be seen later). So once a vertex is assigned a nonzero label, then it can be removed from S1 permanently. (Since its number is greater than 0, it will not be pushed on S1 again.) And S2 is not necessary at all. Thus the following modifications can be done in the algorithm.

All references to S2, including the *while* loop that outputs and transfers the vertices in S2 back to S1, are deleted, and the two occurrences of the statement "if label(w)=0 then label(w) ← c" are replaced by "label(w) ← c". We will call the resulting algorithm L1.

Complexity of algorithm L1: Each vertex v of G is pushed on both S and S1 only once, immediately after v gets a nonzero number. When AF(v), including the newly added entries, is processed completely, v is popped from S. When AT(v) is processed completely, that is, when v is assigned nonzero label, it is popped from S1.

Each entry in the adjacency lists AT and AF is processed exactly once. The number of newly added edges is at most equal to the number of edges in the original graph G.

Thus the complexity of the algorithm is O(n+m), where n is the number of vertices and m is the number of edges of G. Since G is connected $m \geq n-1$, and so the complexity is O(m).

4. ABSTRACTION ALGORITHM

Algorithm A

Input: A 1-sink acyclic graph G = (VG,eG) represented by arrays AT and AF, and labels of VG generated by algorithm L (or L1). The single sink vertex is b.

Output: A directed cut transversal.

Comments: A stack S in which each element is a nonnull set of vertices is used; TOP will denote the top vertex set in S, and SECONDTOP will denote the second (from the top) vertex set in S. "Push {x} on S" means the contents of S are pushed and {x} entered as new TOP. It is assumed that if TOP becomes null, then it is popped automatically. The sets are represented as queues.

Algorithm:

```
begin
  for each x in VG tag(x) ← 0
  tag(b) ← 1
  push {b} on S
  while S is not empty do
    begin
      w ← a vertex in TOP
      i ← label(w)
      while AF(w) is not nil do
        begin
          v ← VTX (AF(w))
```

```
                AF(w) ← LINK (AF(w))
                if tag(v) = 0 and label(v) < i
                  then begin
                         output (v,w)
                         w ← v
                         tag(w) ← 1
                         push {w} on S
                         i ← label(w)
                       end
              end
              [AF(w) is nil]
              while AT(w) is not nil do
                begin
                  v ← VTX (AT(w))
                  AT(w) ← LINK (AT(w))
                  if tag(v) = 0
                    then begin
                           tag(v) ← 1
                           if label(v) = i
                             then TOP ← TOP ∪ {v}
                             else SECONDTOP ← SECONDTOP ∪ {v}
                         end
                end
                [AT(w) is nil]
              TOP ← TOP - {w}
          end
  end
```

The abstraction algorithm, algorithm A, implements the procedure for getting a complete abstraction pair, starting with $(\mathbb{R},\phi)$, where $\mathbb{R}$ is the complete recessional collection generated by algorithm L.

The following has been shown in [8].

4.1 Let $(\mathbf{R}_p,t_p)$ be an abstraction pair for 1-sink acyclic graph G.

(i) If $\mathbb{R}_p \neq \phi$, then $(\mathbf{R}_p,t_p)$ has a successor.

(ii) For each edge (u,v) in t_p, there is a path from sink b to u whose reverse edges (those traversed in the path in the reverse direction) all lie in t_p.

(iii) If $(\mathbb{R}_p,t_p)$ is a complete abstraction pair, then t_p is a directed cut transversal for G.

In algorithm A, to begin with, all vertices other than b have tag 0; b has tag 1. When a vertex is assigned tag 1, it is entered in the stack S. For a vertex w in TOP, if there is an edge (v,w) such that tag(v) = 0 and label(v) < label(w), then (v,w) is output as a transversal edge, v is assigned tag 1 and {v} is pushed on S. If no such edge is found, then all edges (w,v) directed away from w are examined and each v having tag 0 is assigned tag 1 and entered in S, in TOP if label(v) = label(w), and in SECONDTOP if label(v) > label(w): the latter action is due to the facts that (i) if label(w) is i, then (w,v) is in δX_i and SECONDTOP already contains a vertex v' such that for some w' in TOP, (w',v') is an edge, for example, the transversal edge, in δX_i, and (ii) by 3.6, label(v) = label(v'). Then w is deleted from TOP, and another vertex from TOP is taken.

The following can be verified easily.

4.2 For each edge (u,v) output by algorithm A, label(u) < label(v), and so (u,v) is in δX_i for i equal to label(u).

4.3 All vertices in any particular set of S have the same label.

Let Z_i refer to a vertex set of S whose vertices have label i. The elements of Z_i may vary with time.

4.4 The stack S contains Z_j as SECONDTOP and Z_i as TOP at some time if and only if $X_j \supset X_i$ and for each (u,v) in δX_i label(u) = i and label(v) = j.

4.5 When Z_i becomes empty and is popped from S, if the abstraction pair is $(\mathbb{R}_p, t_p)$, then no proper subset of X_i is in $\mathbf{R}_p$; at that time, all vertices in X_i have tag 1.

4.6 When S becomes empty, that is, when Z_k is popped from S, all vertices of G have tag 1.

4.7 Each vertex of G with label i is entered in Z_i exactly once.

4.8 Each Z_i, $1 \leq i \leq k$, appears in S exactly once.

4.9 For each i, $i < k$, an edge in δX_i is output as a transversal edge just before Z_i is entered in S.

4.10 Exactly one edge from each δX_i is output as a transversal edge.

In terms of abstraction pairs, algorithm A starts with $(\mathbb{R}, \phi)$. At some stage, let $(\mathbb{R}_p, t_p)$ be the abstraction pair. Let T represent the set of vertices with tag 1. Then, for each v in T, there is a directed path from y to v where y is either the sink b or the end u' for some edge (u',v') in t_p (or, equivalently, there is a path from b to v whose reverse edges all lie in t_p), and $\mathbb{R}_p$ contains X_i such that no edge from δX_i has been abstracted (or, equivalently, i is not the label of u' for any (u',v') in t_p or, equivalently, no vertex with label i has tag 1). By 4.1(i), if $\mathbb{R}_p \neq \phi$, then a successor is assured. That is, for some i such that X_i is in $\mathbb{R}_p$, there is an edge (u,v) where label(u) is i and v is in T. The algorithm outputs the first such edge (u,v), and the new abstraction pair is $(\mathbb{R}_p - \{X_i\}, t_p \cup \{(u,v)\})$.

When the algorithm terminates, each vertex has tag 1, the abstraction pair $(\mathbb{R}_p, t_p)$ is complete and t_p is a directed cut transversal. Also $|t_p| = |D|$ where D is the disjoint collection of directed coboundaries output by algorithm L (or L1). Hence D is a maximum disjoint collection of directed cuts and t_p a minimum directed cut transversal in G.

Complexity of algorithm A: Each vertex is assigned tag 1 and entered in S once. Each entry of AF and AT is processed once. Thus the complexity of algorithm A is O(m).

Since algorithm A uses only the labels of the vertices, and not the actual vertex sets in $\mathbb{R}$, algorithms L1 and A constitute an optimal algorithm, of complexity O(m), for finding a minimum directed cut transversal in a 1-sink acyclic graph.

REFERENCES

[1] S. Baase, *Computer Algorithms: Introduction to Design and Analysis*, Addison-Wesley, 1978.

[2] R.M. Karp, *Reducibility among Combinatorial Problems* in R.E. Miller and J.W. Thatcher (eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85-103.

[3] C.L. Lucchesi, *A Minimax Equality for Directed Graphs*, Ph.D. thesis, University of Waterloo, Ontario, 1976.

[4] C.L. Lucchesi and D.H. Younger, *A Minimax Theorem for Directed Graphs*, J. London Math. Soc. (2), 17(1978), pp. 1-6.

[5] I.P. McWhirter and D.H. Younger, *Strong Covering of a Bipartite Graph*, J. London Math. Soc. (2), 3(1971), pp. 86-90.

[6] S. Seshu and M.B. Reed, *Linear Graphs and Electrical Networks*, Addison-Wesley, 1961.

[7] K. Vidyasankar, *Minimum Strong Cover in Circuit-Balanced Graphs*, Utilitas Mathematica, 14(1978), pp. 287-303.

[8] D.H. Younger, *From Shortest Paths to Directed Cut Transversals*,

Proceedings of the Second International Conference on Combinatorial Mathematics, New York, 1978.

GENERALIZED K-LEVEL SEARCH REARRANGEMENT

KHALED M. BUGRARA and PAUL W. PURDOM, JR.
Computer Science Department
Indiana University, Bloomington, Indiana 47405.

1. Introduction

A powerful method of efficient searching is called search rearrangement backtracking. It cleverly selects the variable to be investigated next. To make effective use of the method one must have a good way of choosing which variable to introduce at each step. It is often more efficient to introduce the variable with the fewest values. The backtracking with forward checking algorithm of Haralick and Elliot [5], and the *one-level* search rearrangement algorithm of Bitner and Reingold [1] use this approach. The one-level search rearrangement algorithm of Bitner and Reingold tests each unset variable and chooses as the next variable the one with the fewest alternatives; their experimental studies showed that considerable savings are obtainable by using this technique. This algorithm was later analyzed in Brown and Purdom [4], where it was shown that a choice from among the variables with the fewest values results in exponential improvement in the expected running time of the algorithm. They were able to show that for one model of random predicates in CNF with v variables, s literals per clause, and v^{α} clauses in the predicate one-level search rearrangement takes $\exp(\Theta(v^{\frac{s-\alpha-1}{s-2}}))$ time on the average for $s/2 < \alpha < s-1$, and that it takes $\exp(O((\ln v)^{\frac{s-1}{s-2}} v^{\frac{s-\alpha-1}{s-2}}))$ time on the average for $1 < \alpha \leq s/2$. Hence, they have established that search rearrangement, although still exponential, provides exponential improvement over backtracking with no rearrangement [2].

2. The algorithm

An obvious generalization of Bitner and Reingold's backtracking algorithm is based upon considering all k-level extensions of the current node. If, during this preliminary investigation, a variable with no possible values was found, then the values being tested do not occur in any solution, hence, forcing the algorithm to backtrack immediately. Otherwise, after the analysis is completed for all of the remaining variables, the root variable of a subtree of minimum width at level k is selected next [8]. We refer to this algorithm as the k-level search rearrangement algorithm.

3. The model problems

To analyze the average performance of the k-level backtracking algorithm we need a representative problem set. A problem set may be obtained by considering sets of random CNF predicates. In our model a random predicate is one that is the conjunction of t clauses randomly selected from a set of random clauses. A random clause is the disjunction of randomly selected s literals from the set of all possible literals. The uniform probability distribution is assumed over the predicates.

4. The results

We derive lower and upper bound formulas for the expected running time of the algorithm over random sets of satisfiability model problems. We show for a class of random predicates in Conjunctive Normal Form with v variables, s literals in a clause, v^{α} clauses in the predicate, and $1 < \alpha < s - k$, the k-level search rearrangement algorithm takes time

$$\begin{array}{ll} \exp\{\Theta(v^{\frac{s-k-\alpha}{s-k-1}})\}, & 0 \le k \le s-2; \\ O(v^s), & k \ge s-1, \quad \alpha > 1 \end{array}$$

on the average to search for all solutions.

5. Conclusions

For random problem sets, there is a sequence of algorithms (an algorithm for each k) that would allow for exponential improvement in the average running time of large random CNF problems drawn from a distribution with $\alpha > 1$. For $k > s - 2$, the k-level search rearrangement algorithm can solve many sets of random problems in an average time that is polynomial in the size of the problem. This suggests that the subset of problems which require exponential time, when the k-level algorithm is used, is an exponentially small fraction of the total set. The results of the present analysis is an improvement over the earlier analysis of backtracking algorithms [6] for $k = 1$, and it is the first and only one for $k \geq 2$.

The analyzed algorithm requires exponential average time for α less than or equal to one. The reason for this is that there are a large (exponential) number of solutions associated with such problems. However, Goldberg, Purdom, and Brown [4,7] have shown that a simplified version of the Putnam-Davis procedure runs in polynomial average time over many sets of random problems having a large number of solutions.

References

1. James R. Bitner and Edward M. Reingold, "Backtrack Programming techniques", *comm. ACM* **18** (1975), pp. 651–655.
2. Cynthia A. Brown and Paul Walton Purdom, Jr., "An Average Time Analysis of Backtracking", *SIAM J. Comp.* **10** (1981), pp. 583–593.
3. Allen Goldberg, "Average Case Complexity of the Satisfiability Problem", *Proceedings of the Fourth Workshop on Automated Deduction* (1970), pp. 1–6.
4. Allen Goldberg, Paul Walton Purdom, Jr., and Cynthia A. Brown, "Average Time Analyses of Simplified Davis-Putnam Procedures", *Information Processing Letters* **15** (1982), pp. 72–75.
5. Robert M. Haralick and Gordon L. Elliott, "Increasing Tree Search Efficiency for Constrained Satisfaction Problems", *Sixth International Joint Conference on Artificial Intelligence* (1979), pp. 356–364.
6. Paul Walton Purdom, Jr. and Cynthia A. Brown, "An Analysis of Backtracking with Search Rearrangement", *SIAM J. Comp.* **14** (1983), pp. 717–733.
7. Paul Walton Purdom, Jr. and Cynthia A. Brown, "The Pure Literal Rule and Polynomial Average Time", Indiana University Computer Science Dept. Tech. Report No. 128 (June 1982), to appear SIAM J. Comput.
8. Paul Walton Purdom, Jr. Cynthia A. Brown, and Edward L. Robertson "Backtracking with Multi-Level Dynamic Search Rearrangement", *Acta Informatica* **15** (1981), pp. 99–113.

AN APPLICATION OF VOLTERRA SERIES TO THE DESIGN OF NONLINEAR FEEDBACK CONTROLS

SAMIR A. AL-BAIYAT AND MICHAEL K. SAIN
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

ABSTRACT

This paper investigates the application of the Total Synthesis (TS) methods to the design of controller dynamics for nonlinear systems. The TS method was introduced in 1979 [9] and subsequently studied in [10-17]. Here the TS problem is formulated using the Volterra Series in conjunction with the multidimensional Laplace transform. The method is illustrated by designing a control system for a gyro with a single degree of freedom.

I. INTRODUCTION

Volterra series is an elegant and powerful tool for representing the input/output behavior of a nonlinear system. It has a rich history in the engineering literature. In recent years there has been a considerable amount of research devoted to theoretical issues such as existence of Volterra series [3-7] and computation of Volterra kernels [1,2,6,7,8]. However, has been no attempt to use the Volterra series in the design of nonlinear control systems.

The goal of this paper is to apply the Volterra series to the design of nonlinear control systems through the concept of the Total Synthesis Problem (TSP), which deals with the simultaneous design for a specific output response and a reasonable control signal. Much of the early work on the TSP [9,15,16,17] was restricted to the class of linear sytems, where the TSP ideas were formulated using the transfer function. However, later Sain and Peczkowski, in a series of papers [10-14] have extended the TSP ideas to nonlinear systems. The basic idea behind the nonlinear TSP is the linearization of the nonlinear sytem at a sequence of points along an operating line. Then linear design is performed at each point, and the resulting conpensators are scheduled over the entire line of operation.

In this paper the TSP for nonlinear systems will be formulated using the Volterra series. This new fourmulation will enable us to go beyond the zeroth order [13], and first order design.

II. PROBLEM STATEMENT

The Total Synthesis Problem is input/output oriented. In the case of linear systems, the TSP was formulated using the transfer function; one way to extend this idea to nonlinear systems is to use the Volterra series, where now the Volterra kernels play an important role in the design.

To be precise, let $W = L^1([0,T],R)$, $U = L^1([0,T],R)$ and $Y = C([0,T],R)$ denote the spaces of requests to the system, controls to the plant, and responses from the plant, respectively. Let X be an n-dimensional space of bounded differentiable functions. Now consider the single input, single output nonlinear plant

$$\dot{x} = f(x) + g(x)u \qquad x(0) = 0 \qquad f(0) = 0 \tag{1.a}$$

$$y = h(x) \tag{1.b}$$

where $f : X \to X$, $g : X \to X$, and $h : X \to Y$ are assumed to be analytic functions of x in some neighborhood of the free response. It is shown in [6,7] that the nonlinear system (1) admits a Volterra representation provided that the free response has a solution on [0,T]. That is

$$y(t) = \int_0^t p_1(t-\sigma_1)u(\sigma_1)d\sigma_1$$

$$+ \int_0^t\int_0^{\sigma_1} p_2(t-\sigma_1,t-\sigma_2)u(\sigma_1)u(\sigma_2)d\sigma_1 d\sigma_2 + \ldots \tag{2}$$

where $p_i(t-\sigma_1,\ldots,t-\sigma_i)$ is the i-th kernel of the plant.

It is more convenient to have the integral over the range $(0,\infty)$. This is accomplished by defining $p_i(t-\sigma_1,\ldots,t-\sigma_i) = 0$ if any $\sigma_j > t$, $j = 1,2,\ldots,i$. So equation (2) is then:

$$y(t) = \int_0^\infty p_1(t-\sigma_1)u(\sigma_1)d\sigma_1 +$$

$$\int_0^\infty\int_0^\infty p_2(t-\sigma_1,t-\sigma_2)u(\sigma_1)u(\sigma_2)d\sigma_1 d\sigma_2 + \ldots . \tag{3}$$

Let us rewrite (3) using operator notation that is

$$y = \sum_{i=1}^{\infty} P_i[u] = P[u] \tag{4}$$

in which

$$P_i[u] = \int_0^\infty \ldots \int_0^\infty p_i(t-\sigma_1,\ldots,t-\sigma_i)u(\sigma_1) \ldots u(\sigma_i)d\sigma_1 \ldots d\sigma_i . \tag{5}$$

Clearly in (4) the operator $P : U \to Y$ represents the plant. Similarly, if the feedback action of the controller is well defined, then there will be an operator $M : W \to U$ generating control actions from requests and another operator $T : W \to Y$ representing the mapping of requests to the plant responses. For the illustration of these operators see Figure 1.

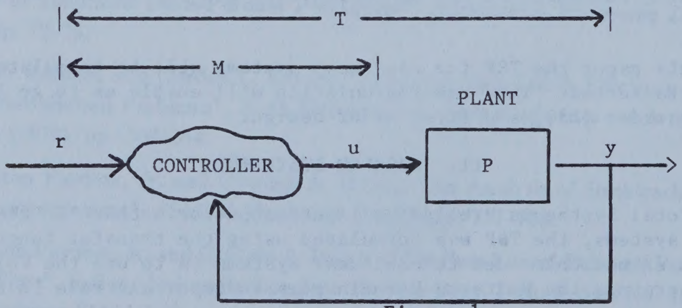


Figure 1.

Now we are in a position to state the Total Synthesis Problem which consists of two steps. The first step is that of the Nominal Design, in which a pair M and T is obtained for a given P, such that the diagram of Figure 2 will commute that is,

$$T = PM . \tag{6}$$

$$\begin{tikzcd} & W \arrow[dl, dashed, "M"'] \arrow[dr, dashed, "T"] & \\ U \arrow[rr, "P"'] & & Y \end{tikzcd}$$

Figure 2.

The second step, Feedback Synthesis, is concerned with realizations of the pair T and M by control dynamics in a specific structure. In this work we adopt the configuration in Figure 3.

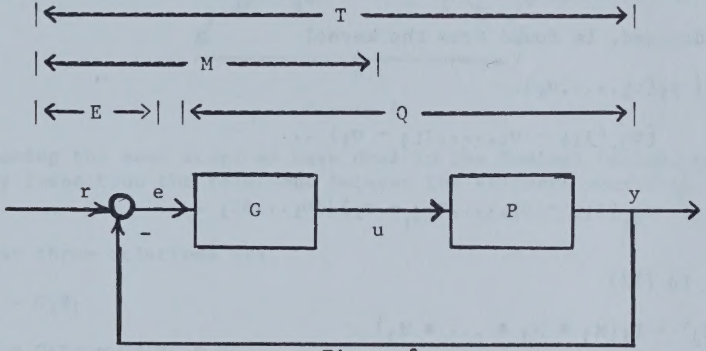


Figure 3.

In Figure 3 the operators $E : W \to V$ represents the mapping of the requests to the errors, given by the space $V = L^1([0,T],R)$, while the operator $G : V \to U$ describes the mapping of errors to the plant controls action.

III. RESULTS

In order to design the operators M, T and ultimately G for a given P, we have to design the Volterra operators associated with them: M_i, T_i, and G_i. Let us start by the designing of M_i and T_i. In the preceding section it was shown that

$$y = P[u] = \sum_{i=1}^{\infty} P_i[u] \; . \tag{7}$$

Clearly it is also possible to write

$$y = T[r] = \sum_{j=1}^{\infty} T_j[r] \tag{8}$$

and

$$u = M[r] = \sum_{k=1}^{\infty} M_k[r] \; . \tag{9}$$

By substituting for u in (7) and equating the result to (8) we get

$$\sum_{j=1}^{\infty} T_j[r] = \sum_{i=1}^{\infty} P_i\Big[\sum_{j=1}^{\infty} M_j[r]\Big] \; . \tag{10}$$

But from the nature of the Volterra operators, for illustration see (5), if r is replaced by cr in (10), where c is an arbitrary constant, then

$$\sum_{j=1}^{\infty} c^{j} T_{j}[r] = \sum_{i=1}^{\infty} P_i\Big[\sum_{j=1}^{\infty} c^{j} M_{j}[r]\Big] . \tag{11}$$

It can be shown that a Volterra operator is a multilinear operator [18-20]. Hence, we can write

$$P_i\Big[\sum_{j=1}^{\infty} c^{j} M_{j}[r]\Big] = \sum_{j_1=1}^{\infty} \cdots \sum_{j_i=1}^{\infty} c^{j_1 + j_2 + \dots + j_i} \cdot$$

$$P_i\big(M_{j_1}[r] \bullet M_{j_2}[r] \bullet \dots \bullet M_{j_i}[r]\big) \tag{12}$$

where the operator $P_i\big(M_{j_1} \bullet M_{j_2} \bullet \dots \bullet M_{j_i}\big)$, note that the argument [r] has been dropped, is found from the kernel

$$\int_{0}^{\infty}\!\!\cdots\!\int p_i(\sigma_1,\dots,\sigma_i)\cdot$$

$$\big(m_{j_1}(\tau_{11} - \sigma_1,\dots,\tau_{1j_1} - \sigma_1) \cdots$$

$$m_{j_i}(\tau_{i1} - \sigma_i,\dots,\tau_{ij_i} - \sigma_i)\big)d\sigma_1\dots d\sigma_i . \tag{13}$$

Note that in (12)

$$P_i[M_j] = P_i\big(M_j \bullet M_j \bullet \dots \bullet M_j\big) . \tag{14}$$

Substituting (12) in (11) and dropping the argument [r] gives

$$\sum_{j=1}^{\infty} c^{j} T_{j} = \sum_{i=1}^{\infty} \sum_{j_1=1}^{\infty} \cdots \sum_{j_i=1}^{\infty} c^{j_1 + \dots + j_i} \cdot$$

$$P_i\big(M_{j_1} \bullet M_{j_2} \bullet \dots \bullet M_{j_i}\big) . \tag{15}$$

Equating like powers of c in both sides of (15) gives

$$T_1 = P_1 M_1 \tag{16}$$

$$T_2 = P_1 M_2 + P_2\big(M_1 \bullet M_1\big) \tag{17}$$

$$T_3 = P_1 M_3 + P_2\big(M_1 \bullet M_2\big) + P_2\big(M_2 \bullet M_1\big) + P_3\big(M_1 \bullet M_1 \bullet M_1\big) \tag{18}$$

$$\vdots$$

$$T_i = P_1 M_i + \sum_{j=2}^{i} \Big\{ \sum_{k_1=1}^{i-j+1} \sum_{k_2=1}^{i-j+2-k_1} \cdots \sum_{k_{j-1}=1}^{i-k_1-k_2-\dots-k_{j-2}-1} \cdot$$

$$P_j\big(M_{k_1} \bullet M_{k_2} \bullet \dots \bullet M_{i-k_1-k_2-\dots-k_{j-1}}\big) \Big\} . \tag{19}$$

The preceding equations (16-19) are convolutional type equations, so it is extremely difficult to design for T and M without going to the transform-domain. But before we go to the transform-domain let us turn to the problem of Feedback Synthesis by redrawing Figure 3 as Figure 4.

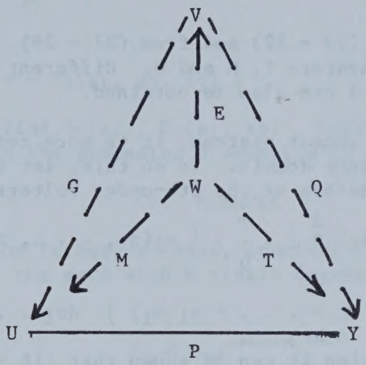


Figure 4.

By following the same steps we have done in the Nominal Design, we can write by inspection the relations between the Volterra operators of M, E, and G.

The first three relations are

$$M_1 = G_1E_1 \tag{20}$$

$$M_2 = G_1E_2 + G_2(E_1 \bullet E_1) \tag{21}$$

$$M_3 = G_1E_3 + G_2(E_1 \bullet E_2) + G_2(E_2 \bullet E_1) + G_3(E_1 \bullet E_1 \bullet E_1) \ . \tag{22}$$

From Figure 3 we have

$$e = r - y \ . \tag{23}$$

Since we can write

$$e = E[r] = \sum_{i=1}^{\infty} E_i[r] \tag{24}$$

and

$$y = T[r] = \sum_{i=1}^{\infty} T_i[r] \ , \tag{25}$$

then (23) can be written as

$$\sum_{i=1}^{\infty} E_i[r] = I[r] - \sum_{i=1}^{\infty} T_i[r] \ . \tag{26}$$

Again, replacing r by cr and equating like powers of c gives

$$E_1 = I - T_1 \tag{27}$$

$$E_2 = -T_2 \tag{28}$$

$$E_3 = -T_3 \,. \tag{29}$$

Clearly, from equations (20 - 22) and from (27 - 29), we can find a direct relation between the operators T, M and G. Different relations between the operators of Figure 4 can also be obtained.

As in the case for linear systems, it is more convenient to perform the design in the frequency domain. To do this, let us define the multidimensional Laplace transform of the ith-order Volterra kernel as [21, 22]

$$P_i(s_1, s_2, \ldots, s_i) \overset{\Delta}{=} \int_0^\infty \cdots \int p_i(\sigma_1, \ldots, \sigma_i)$$

$$\exp\left\{ -(s_1 \sigma_1 + \ldots + s_i \sigma_i) \right\} \, d\sigma_1 \ldots d\sigma_i \,. \tag{30}$$

By applying this definition it can be shown that (16 - 18) can be written in the frequency domain as

$$T_1(s) = P_1(s)\, M_1(s) \tag{31}$$

$$T_2(s_1, s_2) = P_1(s_1 + s_2)\, M_2(s_1, s_2) + P_2(s_1, s_2)\, M_1(s_1)\, M_1(s_2) \tag{32}$$

$$\begin{aligned} T_3(s_1, s_2, s_3) = {} & P_1(s_1 + s_2 + s_3)\, M_3(s_1, s_2, s_3) \\ & + P_2(s_1, s_2 + s_3)\, M_1(s_1)\, M_2(s_2, s_3) \\ & + P_2(s_1 + s_2, s_3)\, M_2(s_1, s_2)\, M_1(s_3) \\ & + P_3(s_1, s_2, s_3)\, M_1(s_1)\, M_1(s_2)\, M_1(s_3) \,. \end{aligned} \tag{33}$$

Now we can perform the design recursively. Since from (31) we can select $T_1(s)$ and $M_1(s)$ for given $P_1(s)$, it is interesting to notice that this is the same design equation given by Peczkowski, et al. [9]. Having selected $T_1(s)$ and $M_1(s)$, we can proceed to (32) where now the only ununknowns are $T_2(s_1, s_2)$ and $M_2(s_1, s_2)$. So again we can select $T_2(s_1, s_2)$ and $M_2(s_1, s_2)$ and we proceed to the next step. In the same way, the higher order kernels are selected.

To complete the design, we have to find the compensator G. To do this let us take the Laplace transform of (20 - 22) and (27 - 29). This gives

$$M_1(s) = G_1(s)\, E_1(s) \tag{34}$$

$$M_2(s_1, s_2) = G_1(s_1 + s_2)\, E_2(s_1, s_2) + G_2(s_1, s_2)\, E_1(s_1)\, E_1(s_2) \tag{35}$$

$$\begin{aligned} M_3(s_1, s_2, s_3) = {} & G_1(s_1 + s_2 + s_3)\, E_3(s_1, s_2, s_3) \\ & + G_2(s_1, s_2 + s_3)\, E_1(s_1)\, E_2(s_2, s_3) \\ & + G_2(s_1 + s_2, s_3)\, E_2(s_1, s_2)\, E_1(s_3) \\ & + G_3(s_1, s_2, s_3)\, E_1(s_1)\, E_1(s_2)\, E_1(s_3), \end{aligned} \tag{36}$$

where

$$\begin{aligned} E_1(s) &= I - T_1(s) \\ E_2(s_1, s_2) &= - T_2(s_1, s_2) \end{aligned} \tag{37}$$

and

$$E_3(s_1, s_2, s_3) = - T_3(s_1, s_2, s_3) \,. \tag{38}$$

Clearly, now we can find $G_1(s)$, $G_2(s_1, s_2)$, $G_3(s_1, s_2, s_3)$ as long as $E_1(s) \neq 0$. Of course the preceding discussion is not limited to the third order kernels.

IV. EXAMPLE

To illustrate the foregoing ideas, consider the problem of designing a control system for the gyro with a single degree of freedom described in Figure 5.

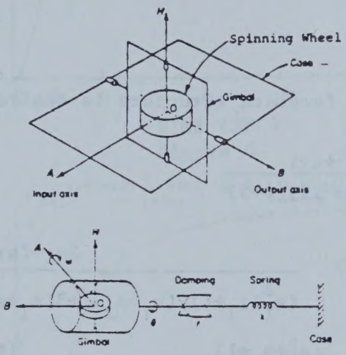


Figure 5.

The equations describing the gyro system are

$$\begin{bmatrix} \dot{x}_1 \\ \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ \\ -\frac{k}{I} x_1 - \frac{f}{I} x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \\ \frac{H}{I} \cos x_1 \end{bmatrix} u \tag{39}$$

and

$$y = x_1, \tag{40}$$

where x_1 is the angle around the output axis, and u is the angle of rotation about the input axis. For simplicity the values of the parameters k, f, I and H are taken to be unity in the simulation. In this example a third degree truncation approximation will be considered by expanding (39) around the origin ($x_o = 0$, $u_o = 0$).

Some calculations lead to the following first three plant kernels:

$$P_1(s) = 1/(s^2+s+1) \tag{41}$$

$$P_2(s_1,s_2) = 0 \tag{42}$$

$$P_3(s_1,s_2,s_3) = \frac{-\,2s_1-(s_1+s_2)-2}{(s_1^2+s_1+1)((s_1+s_2)^3+3(s_1+s_2)^2+6(s_1+s_2)+4)} \cdot \frac{1}{((s_1+s_2+s_3)^2+(s_1+s_2+s_3)+1)} \tag{43}$$

Suppose that the first three kernels of the closed loop response of the gyro were required to be

$$T_1(s) = \frac{1.8}{(s+.4)(s+.9)(s+5)} \tag{44}$$

$$T_2(s_1,\ s_2) = 0 \tag{45}$$

$$T_3(s_1,\ s_2,\ s_3) = 0 \tag{46}$$

Also, assume that a unity feedback structure is desired. Then from (31 - 33) we have

$$M_1(s) = 1.8\ \frac{s^2+s+1}{(s+.4)(s+.9)(s+5)} \tag{47}$$

$$M_2(s_1,\ s_2) = 0 \tag{48}$$

$$M_3(s_1,s_2,s_3) = (1.8)^3\ \frac{2s_1+(s_1+s_2)+2}{(s_1^2+s_1+1)((s_1+s_2)^3+3(s_1+s_2)^2+6(s_1+s_2)+4)} \cdot \frac{(s_1^2+s_1+1)}{(s_1+.4)(s_1+.9)(s_1+5)} \cdot \frac{(s_2^2+s_2+1)}{(s_2+.4)(s_2+.9)(s_2+5)} \cdot \frac{s_3^2+s_3+1}{(s_3+.4)(s_3+.9)(s_3+5)} \tag{49}$$

and from (34 - 36) we have

$$G_1(s) = 1.8\ \frac{s^2+s+1}{s^3+6.3s^2+6.86s} \tag{50}$$

$$G_2(s_1,\ s_2) = 0 \tag{51}$$

$$G_3(s_1,\ s_2,\ s_3) = (1.8)^3\ \frac{2(s_1+1)+(s_1+s_2)}{(s_1^2+s_1+1)((s_1+s_2)^3+3(s_1+s_2)^2+6(s_1+s_2)+4)} \cdot \frac{s_1^2+s_1+1}{s_1^3+6.3s_1^2+6.86s_1} \cdot \frac{s_2^2+\ s_2+1}{s_2^3+6.3s_2^2+6.86s_2} \cdot \frac{s_3^2+s_3+1}{s_3^3+6.3s_3^2+6.86s_3} \tag{52}$$

Computer simulation of the closed loop varifies that the desired response T is achieved. Figures 6 and 7 display the response of the closed loop to step requests of .5 and 1, respectively, and in both cases the nonlinear and the linear controllers are compared with the desired response. It is clear from Figures 6 and 7 that the third order nonlinear controller outperforms the linear one.

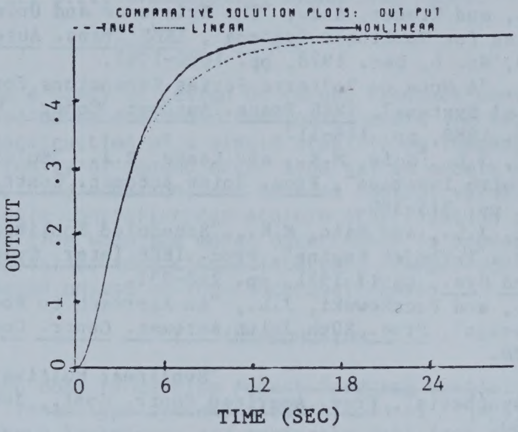


Figure 6.

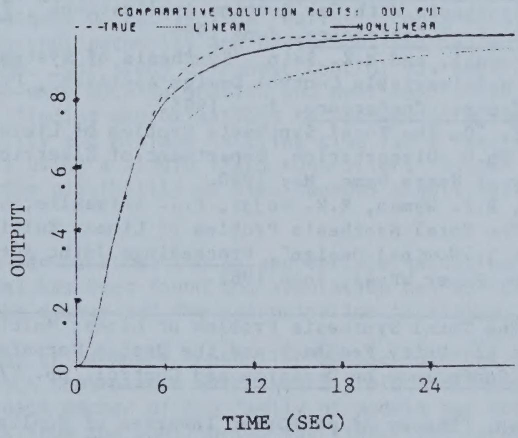


Figure 7.

V. REFERENCES

[1]. Fliess, M., Lamnabhi, M., and Lamnabhi-Lagarrigue, F., "An Algebraic Approach to Nonlinear Functional Expansions", IEEE Trans. Circuits Syst., Vol. CAS-30, No. 8, August 1983, pp. 554-570.

[2]. Normand-Cyrot, D., "An Algebraic Approach to the Input/Output Description of Nonlinear Discrete-Time Systems", Proc. American Contr. Conf., June 1982, pp. 466-471.

[3]. Sandberg, I.W., "On Volterra Expansions for Time-Varying Nonlinear Systems", IEEE Trans. Circuits Syst., Vol. CAS-30, No. 2, February 1983, pp. 61-67.

[4]. Brockett, R.W., "Convergence of Volterra Series on Infinite Intervals and Bilinear Approximations", in Nonlinear Systems and Applications, V. Lakshmikantham, Ed., Academic Press, 1977, pp. 39-46.

[5]. Gilbert, E.G., "Functional Expansions for the Response of Nonlinear Differential Systems", *IEEE Trans. Automat. Contr.*, Vol. AC-22, No. 6, December 1977, pp. 909-921.

[6]. Brockett, R.W., "Volterra Series and Geometric Control Theory", *Automatica*, Vol. 12, 1976, pp. 167-176 (addendum with E. Gilbert, Vol. 12, pp. 635).

[7]. Lesiak, C., and Krener, A.J., "The Existence and Uniqueness of Volterra Series for Nonlinear Systems", *IEEE Trans. Automat. Contr.*, Vol. AC-23, No. 6, Dec. 1978, pp. 1090-1095.

[8]. Fliess, M., "A Note on Volterra Series Expansions for Nonlinear Differential Systems", *IEEE Trans. Automat. Contr.*, Vol. AC-25, No. 1, Feb. 1980, pp. 116-117.

[9]. Peczkowski, J.L., Sain, M.K., and Leake, R.J., "Multivariable Synthesis with Inverses", *Proc. Joint Automat. Contr. Conf.*, June 1979, pp. 375-380.

[10]. Peczkowski, J.L., and Sain, M.K., "Scheduled Nonlinear Control Design for a Turbojet Engine", *Proc. IEEE Inter. Symposium on Circuit and Sys.*, April 1981, pp. 248-251.

[11]. Sain, M.K., and Peczkowski, J.L., "An Approach to Roubust Nonlinear Control Design", *Proc. 20th Joint Automat. Contr. Conf.*, June 1981, paper FA-3D.

[12]. ______________________________, "Nonlinear Multivariable Design By Total Synthesis", *Proc. American Contr. Conf.*, June 1982, pp. 252-260.

[13]. ______________________________, "Nonlinear Control By Coordinated Feedback Synthesis With Gas Turbine Applications", *Proceedings American Control Conference*, June 1985.

[14]. J.L. Peczkowski, and M.K. Sain, "Synthesis of System Responses: Nonlinear Multivariable Control Design Approach", *Proceedings American Control Conference*, June 1985.

[15]. R.R. Gejji, "On the Total Synthesis Problem of Linear Multivariable Control", Ph.D. Dissertation, Department of Electrical Engineering, University of Notre Dame, May 1980.

[16]. M.K. Sain, B.F. Wyman, R.R. Gejji, P.J. Antsaklis, and J.L. Peczkowski, "The Total Synthesis Problem of Linear Multivariable Control, Part I: Nominal Design", *Proceedings Joint Automatic Control Conference*, Paper WP-4A, June 1981.

[17]. ____, ______, "The Total Synthesis Problem of Linear Multivariable Control, Part II: Unity Feedback and the Design Morphism", *Proceedings 20th IEEE Conference on Decision and Control*, pp. 875-884, December 1981.

[18]. M. Schetzen, "Theory of pth-Order Inverses of Nonlinear Systems", IEEE *Trans. Circuits and Systems*, Vol. CAS-23, No. 5, pp. 285-291, May 1976.

[19]. W.J. Rugh, *Nonlinear System Theory: The Volterra/Wiener Approach*, Baltimore, The Johns Hopkins University Press, 1981.

[20]. W.A. Porter, "An Overview of Polynomic System Theory", *IEEE Proceedings*, Vol. 64, No. 1, pp. 18-23, January 1976.

[21]. J.J. Bussgang, L. Ehrman, and J. Graham, "Analysis of Nonlinear Systems with Multiple Inputs", *IEEE Proceedings*, Vol. 62, No. 8, pp. 1088-1119, August 1974.

[22]. E. Bedrosian, and S.O. Rice, "The Output Properties of Volterra Systems (Nonlinear Systems with Memory) Driven by Harmonic and Gaussian Inputs", *IEEE Proceedings*, Vol. 59, No. 12, pp. 1688-1707, December 1971.

DESIGN OF A COMMON FEEDBACK CONTROLLER FOR A FAMILY OF NONLINEAR SECOND ORDER SYSTEMS†

G. Gu and E. B. Lee
University of Minnesota, Dept. of Electrical Engineering
123 Church St. S.E., Minneapolis, Minnnesota 55455

ABSTRACT

Results from a study of design of robust feedback controllers for systems described by second order nonlinear models are reported. Studied in detail was construction of a single stabilizing feedback controller for a parametrized family of second order type system models with hard spring nonlinearity (Duffing equation) and bounded control values. It has been found that a single controller can achieve stabilization and approximate minimum settling time when the model uncertainty is represented by a parameter taking values in a closed interval. Comparison to an optimal controller is based on use of the isochronal curves.

I. INTRODUCTION

Much is being done to develop robust feedback controller design techniques for linear type systems (see references [1]-[5] for example). In most cases these techniques are seriously deficient when applied to, for example, design of controllers for disk (motor) drives of computer machines. The linear techniques do not include realistic constraints such as saturation of control authority and basic nonlinear effects within the controlled system. Our study of robust control has been directed at seeing what can be done when such saturation and nonlinear effects are admitted; specifically at finding approximations to bang-bang feedback control for achieving stability and minimum settling time for a family of second order nonlinear models using a single fixed controller. Desoer and Lin [6] have also looked at the possibility of simultaneous stabilization of nonlinear systems.

A technique for constructing an appropriate switching locus for this family of systems has been found and simulation has verified the stability robustness of the design and the approximation to minimum settling time.

The technique of feedback controller design involves looking at the family of feedback controllers which would yield a minimum settling time controller for each member of the family of models and noting that for small deviations from the equilibrium the linearized solutions are each similar and that for large deviations from the equilibrium an average (smoothed version) of the values of the feedback controller leads to a stabilizing controller which also provides an approximation to minimum settling time.

This presentation will cover some of our results that depend on previous work on time optimal control and robust control system synthesis techniques.

The family of systems which we seek a common feedback controller for are described by the Duffing equation (second order ordinary differential equation) with a parameter ε and control function u(t):

$$\ddot{x} + x + \varepsilon x^3 = u$$

---

† Research supported by the National Science Foundation under grant number DMS-8413129.

In the study we have let $0 \leq \varepsilon \leq 10$, and imposed the controller saturation restraint $|u| \leq 1$. Other nonlinear second order families of systems depending on a parameter can be studied in a similar manner; in particular results for a controlled pendulum [7] can be similarly derived.

In the next section we will provide an overview of time optimal feedback controller construction for second order systems as described by the Duffing equation with a parameter ε. For each fixed value of $\varepsilon \in [0,10]$ the construction of the switching curve that gives the optimal feedback controller is presented. Comparisons of the results of using each member of the family of optimal feedback controllers helps one select a common feedback controller that is stabilizing and approximates time optimal control for the whole family of systems as described in section III.

II. TIME OPTIMAL CONTROL AND THE SWITCHING LOCUS

A comparison of time optimal feedback controllers (which "steer" to $x=0$, $\dot{x}=0$) for a family of systems specified by Duffing's equation

$$\ddot{x} + x + \varepsilon x^3 = u(t), \quad 0 \leq \varepsilon \leq 10, \quad |u(t)| \leq 1 \tag{1}$$

will be done in this section.

For time optimal control to the origin $x=0$, $\dot{x}=0$, consider finding an optimal feedback (control) function $u = \Psi(x,\dot{x})$ for the family of systems (1) for each fixed ε with $0 \leq \varepsilon \leq 10$ (it is assumed that both $x(t)$ and $\dot{x}(t)$ are available). Rewrite equation (1) as the system of equations

$$\text{S)} \quad \begin{aligned} \dot{x} &= y \\ \dot{y} &= -x - \varepsilon x^3 + u(t). \end{aligned}$$

To find $\Psi(x,y)$ we have to find the "switch curve" W [7]. From [7], and [8], we have the conclusion that the phase plane ((x,y)-plane)) is divided into two parts by the switch curve W. For the upper part, function Ψ has value -1, and for the lower part $\Psi = +1$. It is not trivial to find the switch curve W since system (1) is nonlinear. Also, the adjoint system equation and the system equation (differential equations for x and y) which can be used to construct W are coupled together. Thus in general, one can't find a closed form expression for the switch curve.

The construction of the time optimal feedback function $\Psi(x,y)$ for the Duffing equation is described in detail in the Lee-Markus text [7] (when the parameter $\varepsilon = 0$ see example 3 page 11, and when the parameter $\varepsilon = 2$ see example 1 page 444). For the sake of completeness we will describe the construction of the time optimal feedback function $\Psi(x,y)$ by describing the construction of a switching curve W which divides the (x,y)-plane into two regions such that $\Psi(x,y) \equiv -1$ "above" the curve W and $\Psi(x,y) \equiv +1$ "below" the curve for each fixed ε. A similar situation will prevail in the case of finding a common feedback controller for the whole family of systems.

For time optimal control to the origin (0,0) in the (x,y)-plane with the restraint $|u(t)| \leq 1$ it is found that $u(t) \equiv +1$ or $u(t) \equiv -1$ on various intervals of time. Thus consider the two families of curves in the (x,y)-plane as generated by solutions of the ODE's with $u(t) \equiv +1$ or $u(t) \equiv -1$:

$$\text{S}_{\pm}) \quad \begin{aligned} \dot{x} &= y \\ \dot{y} &= -x - \varepsilon x^3 \pm 1. \end{aligned}$$

Now for each fixed $\varepsilon \in [0,10]$ every point of the (x,y)-plane can be connected to the origin by picking successive paths only from these +,- families of curves, assuring existence of an optimal controller.

Also, the intervals of time that the solution curves of the S_+ or S_- families are followed for time optimal control to the origin are determined by successive zero crossings of the solution component $\eta_2(t)$ of the adjoint system of ODE's:

$$\text{A)} \quad \begin{aligned} \dot{\eta}_1 &= \eta_2(1+3\varepsilon[x(t)]^2) \\ \dot{\eta}_2 &= -\eta_1 \, . \end{aligned}$$

which are dependent on the availability of $x(t)$ from either S_+ or S_-.

To construct the switching curve $W = W(\varepsilon)$ let $W_+{}^1$ be the solution of S_- through the origin and lying in the fourth quadrant of the (x,y)-plane (see figure 1) and let $W_-{}^1$ be the solution of S_+ through the origin and lying in the second quadrant. In figure 1 the curves $W_+{}^1 = W_+{}^1(\varepsilon)$ for various values of ε between 0 and 10 are displayed. In particular

$$W_+{}^1: \quad y = -(2x-x^2 - \frac{\varepsilon}{2} x^4)^{1/2} \quad \text{for } 0 \leq x \leq x_1$$

$$W_-{}^1: \quad y = (-2x-x^2 - \frac{\varepsilon}{2} x^4)^{1/2} \quad \text{for } -x_1 \leq x \leq 0$$

$$\text{where } x_1 = x_1(\varepsilon) = \left(\frac{2}{\varepsilon} + \sqrt{(\frac{2}{\varepsilon})^2 + (\frac{2}{3\varepsilon})^3} \right)^{1/3} + \left(\frac{2}{\varepsilon} - \sqrt{(\frac{2}{\varepsilon})^2 + (\frac{2}{3\varepsilon})^3} \right)^{1/3} .$$

Typical values of x_1 are

$$x_1 = \begin{array}{ll} 0.646835093 & \text{for } \varepsilon = 10 \\ 1.179509026 & \text{for } \varepsilon = 1 \\ 1.962224018 & \text{for } \varepsilon = 0.01 \\ 1.999600240 & \text{for } \varepsilon = 0.0001 \, . \end{array}$$

Clearly, the first two values coincide with those in reference [8], and as $\varepsilon \to 0$, $x_1 \to 2$ which is same as in [7].

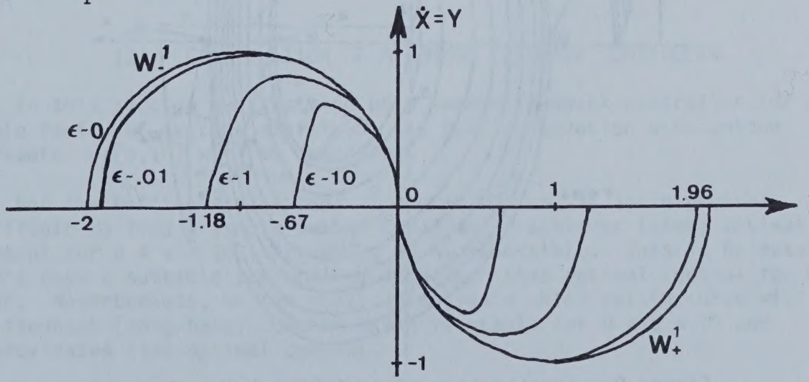


Figure 1: Initial lobe of switching curve

It seems impossible to find an explicit expression for the other lobes of switch curve, because to solve for the other lobes, we have to solve the adjoint system of equations A) as given above with $\eta_2(0) = 0$, $\eta_1(0) = +1$.

One way to get out of the difficulties is using numerical techniques to solve for x and η to find the other lobes of the switch curve. Again it is noted that the switch curve is anti-symmetric, so we need only to solve the u=+1 case (or u=-1 case). It is easy to find an algorithm to find the switch curve for $|x|<M$ for a given domain of interest as given by M.

To see how this algorithm proceeds consider the initial lobe of the switching curve W_-^1 as described above and displayed in figure 1. We next reflect W_-^1 along solutions of S_+. From points labeled 1, 2, ..., 9 of figure 2 follow backwards in time along the corresponding solutions of S_+ for a time duration equal to the interval between successive zeros of $\eta_2(t)$. That is from say point 1 with $\eta_1(0) = 1$, $\eta_2(0) = 0$ follow S_+ and consider the solution component $\eta_2(t)$ of the adjoint equation until $\eta_2(t)$ is zero again. This determines a point 1a on W_+^2. We repeat this for each of the remaining points 2, 3, ..., 9 to obtain the corresponding points 2a, 3a, ..., 9a of W_+^2 (see figure 2). Once W_+^2 is generated its reflection through a conjugate interval along solution curves of S_- provides the portion of the switch curve labeled W_-^3. The remaining portions of the switch curve are found in a similar manner.

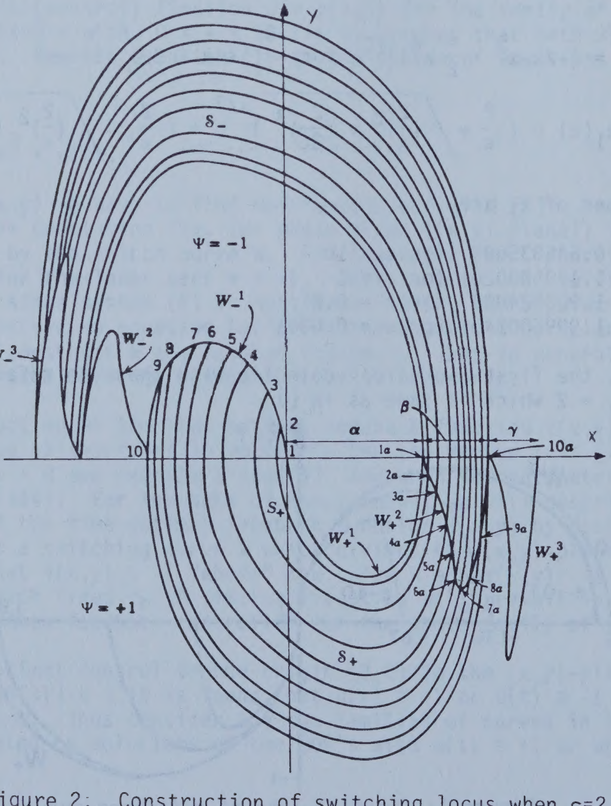
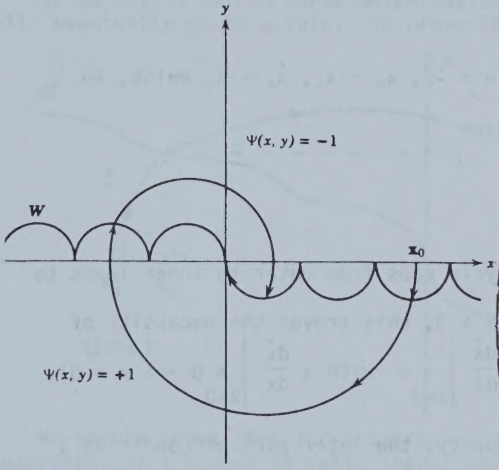


Figure 2: Construction of switching locus when $\varepsilon=2$.

Figure 2 shows the construction of the switching curve W when the parameter value $\varepsilon = 2$. Figure 3 is switching locus for the special case when $\varepsilon = 0$. In figure 4 we have constructed the isochronal curves for steering to the origin. The interior and each of these curves correspond to the attainable set, going backwards in time from the origin during the time interval $[0,T]$, where $T > 0$ is the value of the isochrone. In figure 4 we have constructed several different isochronal curves for the case when $\varepsilon = 1$. The isochrones will provide a basis for assessing the quality of a feedback controller designed for the family using a comparison at each value of ε (essentially a way of constructing a min-max controller).

Note that each time optimal feedback controller for various values of the parameter is a stabilizing controller in the sense that each solution $(x(t), y(t))$ goes to $(0,0)$ in finite time. We will require that the common feedback controller for the family of nonlinear system described by the duffing equation with parameter $\varepsilon \in [0,10]$ also have this property.



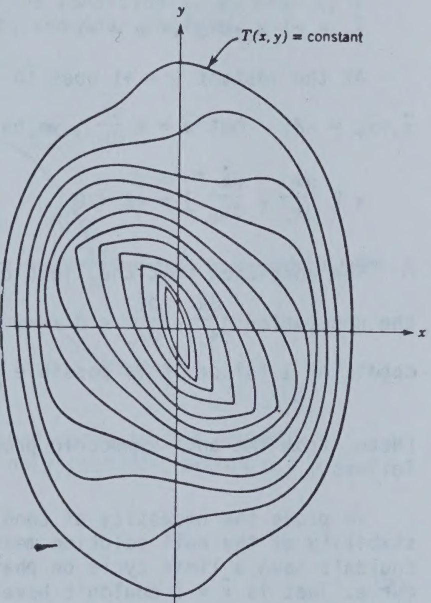


Figure 3: Switching locus when $\varepsilon=0$.

Figure 4: Isochronal curves $(\varepsilon=1)$.

III. CONSTRUCTION OF A COMMON FEEDBACK CONTROLLER

In this section construction of a common feedback controller for the whole family of systems satisfying the Duffing equation with unkown parameter $\varepsilon \in [0,10]$ will be described.

For the Duffing's equation: $\ddot{x} + x + \varepsilon x^3 = u$, it is in general difficult to find a single switch curve which achieves (time) optimal control for $0 \leqslant \varepsilon \leqslant 10$. Actually, it is impossible. This is because we don't have a suitable performance index for time optimal control for $0 \leqslant \varepsilon \leqslant 10$. Nevertheless, we can still investigate which switch curve will lead to feedback (bang-bang) control which is stable for $0 \leqslant \varepsilon \leqslant 10$ and approximates time optimal control.

The following theorem is the main result on construction of a stabilizing switch curve.

Theorem: A necessary and sufficient condition for a stable switch curve is: (assumption: a switch curve, if written as $\dot{x} = f(x)$ then f is continuous).

1) The switch from $u = +1$ to -1 only takes place at $\dot{x} \geqslant 0$; similarly, the switch from $u = -1$ to $+1$ only takes place at $\dot{x} \leqslant 0$;

2) The switch curve is inside the second and fourth quadrants.

Proof: necessity: first we should note that an equivalent expression for Duffing's equation is $\dot{x}^2 = 2ux - x^2 - \frac{\varepsilon}{2} x^4 + C$ which defines a limit cycle on the phase plane. Therefore, if the switch curve leads to a stable feedback system, it must be such that, every time the control switches, the trajectory of x-$\dot{x}$ should go to the inner limit cycle, not outer or remain on same limit cycle. Consider

$$\ddot{x}_- = -1-x_--\varepsilon x_-^3\text{:}\quad u = -1$$

$$\ddot{x}_+ = +1-x_+-\varepsilon x_+^4\text{:}\quad u = +1$$

At the instant $u = +1$ goes to $u = -1$, $x_+ = x_-$, $\dot{x}_+ = \dot{x}_-$ exist, so $\ddot{x}_- - \ddot{x}_+ = -2$; but $\ddot{x} = \dot{x} \frac{d\dot{x}}{dx}$, we have:

$$\dot{x}\left[\frac{d\dot{x}}{dx_-} - \frac{d\dot{x}}{dx_+} \right] = -2 < 0$$

The condition that the limit cycle goes from outer to inner leads to the conclusion $\frac{d\dot{x}}{dx_-} - \frac{d\dot{x}}{dx_-} < 0 \iff \dot{x} > 0$, this proves the necessity of condition 1 (since it is possible $\left.\frac{d\dot{x}}{dx}\right|_{\dot{x}=0} = \infty$ with $\left.\dot{x}\frac{d\dot{x}}{dx}\right|_{\dot{x}=0} \neq 0 \Longrightarrow \dot{x} \geqslant 0$).

(Note, from the anti-symmetric property, the later part of condition 1 follows.)

To prove the necessity of condition 2, we should note that the stability of the null solution means for either $u = 1$ or $u = -1$, we couldn't have a limit cycle on phase plane without crossing the switch curve. That is $\dot{x} = 0$ couldn't have two solutions for x which have same signs for $u = 1$ or $u = -1$. But

$$\dot{x}^2 = 2x - x^2 - \frac{\varepsilon}{2} x^4 + C \quad \text{for } u = 1$$

If $x = x_0$ is one solution, then $C = x_0^2 + \frac{\varepsilon}{2} x_0^4 - 2x_0$ exists

$$\therefore\ \dot{x} = 0 \Longrightarrow 2(x-x_0)-(x-x_0)(x+x_0)- \frac{\varepsilon}{2} (x-x_0)(x^2+x_0^2)(x-x_0) = 0$$

Let $x_1 \neq 0$, s.t. $\dot{x} = 0$, then we have (there is only one real solution)

$$2 - (x_1+x_0) - \frac{\varepsilon}{2} (x_1^2+x_0^2)(x_1+x_0) = 0$$

or: $(x_1+x_0)[1 + \frac{\varepsilon}{2} (x_1^2+x_0^2)] = 2$ for $u = 1$

That is that x_1 has same sign as x_0 is only possible for x_0, x_1 in first quadrant ($x_1 > 0$, $x_0 > 0$).

Same procedure, we have:

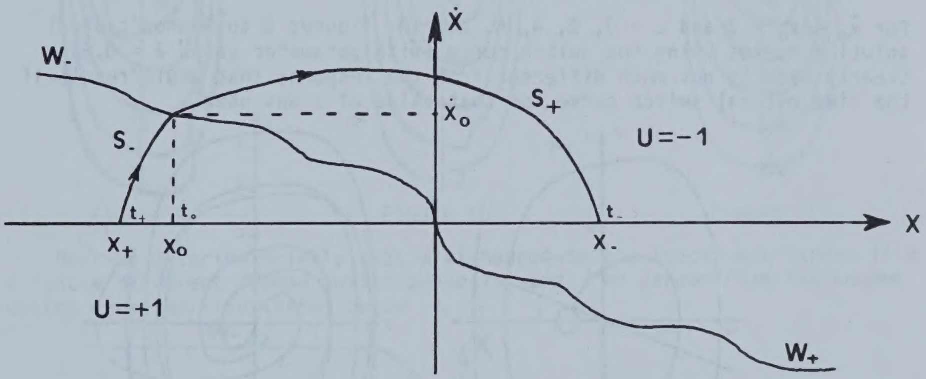
$$(x_1+x_0)[\, 1+ \frac{\varepsilon}{2}\,(x_1{}^2+x_0{}^2)\,] = -2 \quad \text{for } u = -1$$

which is possible for x_1 and x_0 having same sign only on third quadrant ($x_1 < 0$, $x_0 < 0$).

Because our system is stable, these limit cycles couldn't exist. That is whenever x, $\dot{x}$ on first quadrant, could only give control $u = -1$, and whenever x, $\dot{x}$ on third quadrant, could only give control $u = 1$. It is equivalent to saying the switch curve is only in second and fourth quadrants. This proves the necessity of condition 2. {It should be mentioned that the switch curve must pass through origin to make the proof more clear, but it is obvious.}

Sufficiency:

If we have a switch curve which satisfies condition 1, 2, then x, $\dot{x}$ will eventually go to origin. To prove it, consider the plot:



We should prove that for arbitrary switch curve in 2nd and 4th quadrants, $|x_-| < |x_+|$ is valid (here, we only consider switch $u = 1 \rightarrow u = -1$, the other case is similar). We have:

$$\dot{x} = 0 \implies \begin{cases} 2x_+-x_+{}^2- \frac{\varepsilon}{2}\, x_+{}^4 + C_+ = 0 \\ -2x_--x_-{}^2- \frac{\varepsilon}{2}\, x_-{}^4 + C_- = 0 \end{cases} \tag{2}$$

$$(2) \implies \quad 2\, x_++2x_- -(x_+{}^2-x_-{}^2)- \frac{\varepsilon}{2}\,(x_+{}^4+x_-{}^4) = 4x_0$$

or:

$$(x_+{}^2-x_-{}^2)[1+ \frac{\varepsilon}{2}\,(x_+{}^2+x_-{}^2)] = 2(x_++x_-)-4x_0 = 2[(x_--x_0)-(x_0-x_+)] \tag{3}$$

Clearly, x_0 and x_+ have the same sign because the switch curve is only in 2nd and 4th quadrants. Recognize that x_--x_0 is the distance between x_- and x_0, let it be d_2, x_0-x_+ is the distance between x_0 and x_+, let it be d_1. Now, suppose $|x_-| < |x_+|$ is not valid, that is $|x_-| > |x_+|$ is assumed. Then, the left hand of equation (3) < 0. But, from the proof of necessity of condition 2, the two roots have different signs if switch curve passes through only 2nd and fourth quadrants. This leads to the conclusion $d_2 > d_1$. That is, the right side of eqn. (3) > 0, which contradicts the assumption $|x_-| > |x_+|$. This proves the sufficiency of the condition.

As an example, a switch curve in 1st and 3rd quadrants is constructed and simulated by computer to show a limit cycle. The switch curve is

$\dot{x} = \frac{1}{2}x$ and initial conditions are $x_0 = \dot{x}_0 = 2$ with $\varepsilon = 2$ for the Duffing's equation (see fig. 5).

As mentioned earlier, it is difficult to define a better or worse common switch curve for $0 \leqslant \varepsilon \leqslant 10$. But an intuitive guess is that a switch curve adopts the optimal switch curve for $\varepsilon = 0$ near the origin and a straight line elsewhere is better (because it is locally optimal). It is simulated by the following switch curve W:

$$\dot{x} = \begin{cases} \sqrt{1-(1-0.5)^2} & \text{for } -\infty < x \leqslant -0.5 \\ \sqrt{1-(1+x)^2} & \text{for } -0.5 < x \leqslant 0 \\ -\sqrt{1-(1-x)^2} & \text{for } 0 < x \leqslant 0.5 \\ -\sqrt{1-(1-0.5)^2} & \text{for } 0.5 \leqslant x \leqslant \infty \end{cases} .$$

For $\dot{x}_0 = x_0 = 2$ and $\varepsilon = 0$, 2, 4, 6, and 10, figures 6 to 8 show typical solution curves using the switch curve (with parameter value $\delta = 0.5$). Clearly each is not much different from the response that would result if the time optimal switch curve for that value of ε was used.

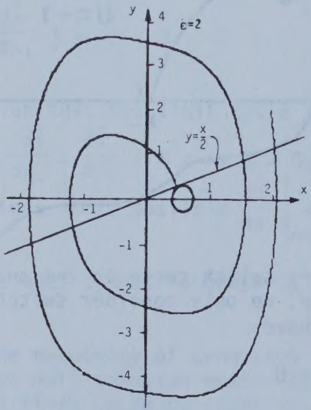


Figure 5

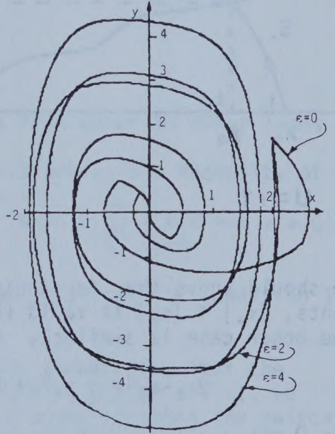


Figure 6

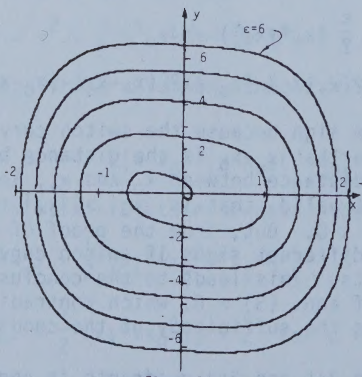


Figure 7

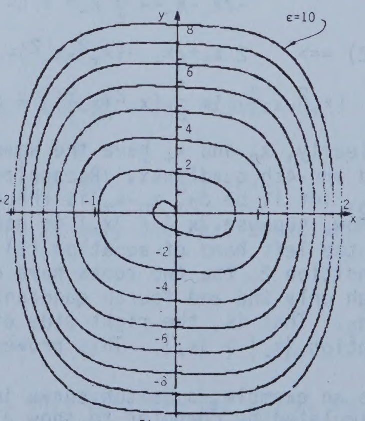


Figure 8

To get some feel for the approximation to time optimal control which is obtained when the common switch curve W (with δ = 0.5) is used the isochronal curves as defined earlier were constructed for the optimal switch curve with parameter value ε = 5 as shown in figure 9. These curves can be compared with the corresponding isochronal curves when the common switch curve W is used (at the parameter value ε = 5) see figure 10. Figure 11 corresponds to a slightly different switch curve with (δ= .5).

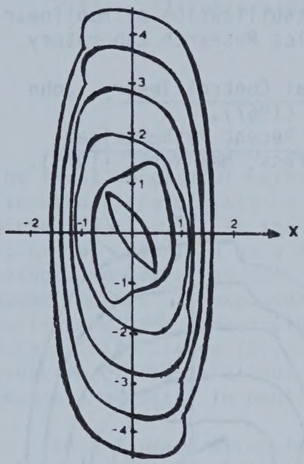


Figure 9

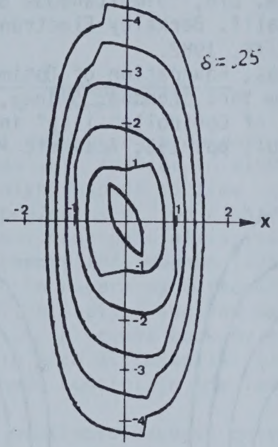


Figure 10

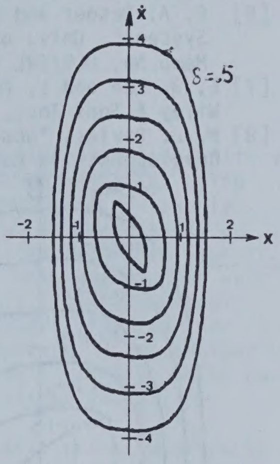
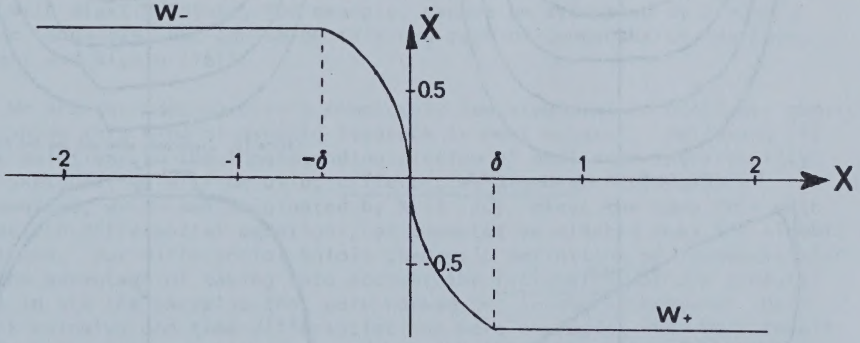


Figure 11

Next we describe briefly what will happen to the isochronal curves if a slightly different common switch curve is used. We parametrize the common switch curve as illustrated below



The sequence of figures 12-15 show what happens to the isochronal curves for the fixed parameter value ε = 5 when δ is changed from 0.5 to 0.25. Further study is now underway to see which value of δ is the "best" choice for each value of ε in the range from 0 to 10.

REFERENCES

[1] J. B. Cruz and W. R. Perkins, "A New Approach to the Sensitivity Problem in Multivariable Feedback System Design", IEEE Trans. on Automat. Contr., AC-9 (1964), pp. 216-223.

[2] G. Zames and B. A. Francis, "Feedback, Minimax Sensitivity, and Optimal Robustness", IEEE Trans. Automat. Contr., AC-28 (1983), pp. 585-601.

[3] C.-C. Chu, J. Doyle and E. B. Lee, "Optimal Feedback Control with Model Uncertainty", submitted to Int. J. Control.

[4] J. Ackermann and S. Turk, "A Common Controller for a Family of Plant Models", Proc. IEEE CDC (1982), pp. 240-247.

[5] A. Pascoal and P. Khargonekar, "Remarks on Weighted Sensitivity Minimization", Proc. Twenty-Third Annual Allerton Conf. on Comm. Contr. and Comp. (1985).

[6] C. A. Desoer and C. A. Lin, "Simultaneous Stabilization of Nonlinear Systems", Univ. of Calif. Berkeley Electronics Research Laboratory Memo No. UCB/ERL M82/79, 1982.

[7] E. B. Lee and L. Markus, Foundation of Optimal Control Theory, John Wiley & Sons Inc., New York, London, Sydney, (1967).

[8] M. J. Davies, "Domain of Controllability" in Recent Mathematical Developments in Control, pp. 153, Academic Press, New York, (1973).

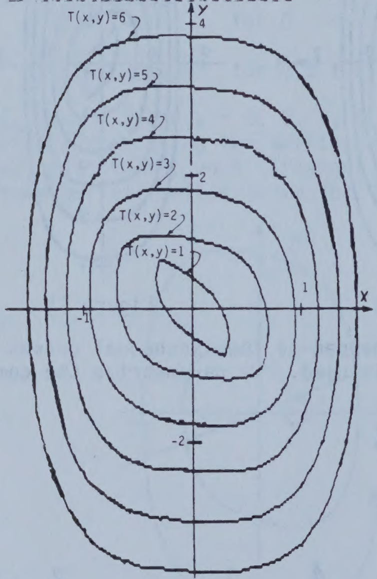


Figure 12: Isochrones when ε=5 and δ=0.5.

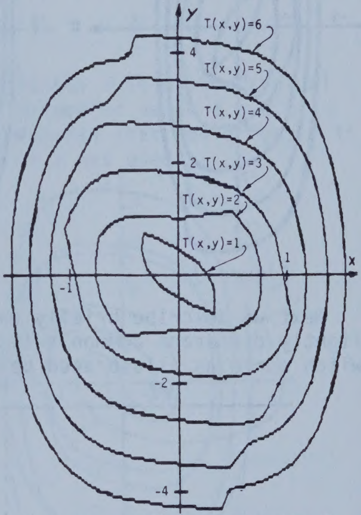


Figure 13: Isochrones when ε=5 and δ=0.25.

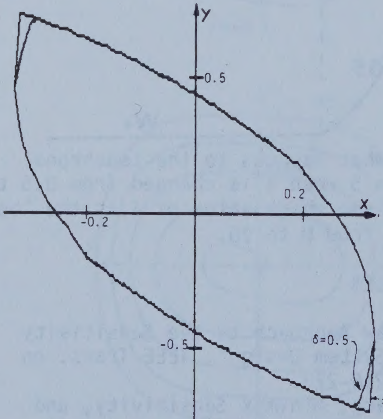


Figure 14: Comparison of T=1 Isochrones at δ=0.5 and δ=0.25 (ε=5).

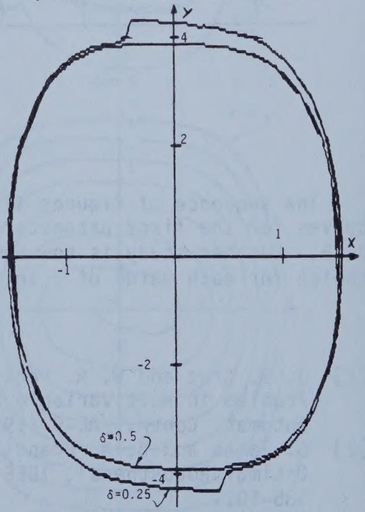


Figure 15: Comparison of T=6 Isochrones

A NEW APPROACH TO THE NONINTERACTING CONTROL PROBLEM IN NONLINEAR SYSTEMS THEORY*

Michel Fliess**
Coordinated Science Laboratory
University of Illinois
1101 W. Springfield Ave.
Urbana, Illinois 61801

INTRODUCTION

Nonlinear input-output decoupling problems have already a long history. The first successful methods were related to eliminating variables using iterated differentiations with respect to time (Porter [19], Freund [5], Sinha [25], ...). At the beginning of the eighties a new trend appeared. It can be described as a nonlinear generalization of the so-called geometric methods in linear systems theory (cf. Wonham [27]). This approach is very appealing and was expounded in papers by Hirschorn [7], Isidori, Krener, Gori-Giorgi and Monaco [10], Nijmeijer and van der Schaft [17], van der Schaft [21], Claude [2], ... A recent book by Isidori [8] gives a clear overview of its relationship with differential geometric methods, which have become a mainstay in nonlinear control in the last fifteen years.

In all the previously mentioned papers, only static state-feedback is considered. As for linear systems [27], it is easy to see that with such a feedback many natural decoupling problems cannot be solved. Singh [24], Descusse and Moog [3], Isidori [8,9] showed recently that a peculiar dynamical feedback accomplishes the decoupling. It consists of a finite set of integrators followed by an ordinary static state-feedback. Robot arms with elastic joints, for example, cannot be decoupled by ordinary static feedbacks, but can be by this new type of compensation (de Luca, Isidori and Nicolo [16]).

We are introducing here a completely new viewpoint in nonlinear control, that shows this kind of dynamic feedback is most natural. Following [4] where solutions to the long-standing problem of nonlinear invertibility were sketched, we will be using differential algebra. This area of mathematics, which was originated by Ritt [20], plays the same role with respect to differential equations, as commutative algebra does for algebraic equations. Our differential Galois theoretic definition of feedbacks also has the advantage of taking into account the rationality of the computations in all the examples that were worked out in the literature: Only matrix calculus and time-differentiations were employed. Our main result is the following: the noninteracting control problem can be solved by extended feedback if, and only if, the system is differentially dominant, i.e., right-invertible. This is a generalization of a well-known result for linear systems, where the noninteracting control problem can be solved if, and only if, the transfer matrix is right-invertible (cf. Wolovich [26]).

---

*This work was supported by the Joint Services Electronics Program under Contract N00014-84-C-0149.

**The author is with the Laboratoire des Signaux et Systèmes, C.N.R.S.-E.S.E., Plateau du Moulon, 91190 Gif-sur-Yvette, France.

More general decoupling problems including perturbations rejection can be solved along the same line. Details will be given elsewhere.

A. A Brief Overview of the Noninteraction Control Problem via Static State-Feedback (cf. Isidori [8])

Consider the following control-linear system

$$\left\{\begin{array}{l} \dot{q}(t) = A_0(q) + \sum_{i=1}^{m} u^i(t) A_i(q) \\ \\ y(t) = h(q) \end{array}\right. \tag{1}$$

The state q evolves on a finite-dimensional $\mathbb{R}$-analytic manifold Q. The vector fields $A_0, A_1, \ldots, A_m : Q \to TQ$ (tangent bundle) and the ouptut map $h = (h^1, \ldots, h^\ell) : Q \to \mathbb{R}^\ell$ are $\mathbb{R}$-analytic. The control-vector $u = (u^1, \ldots, u^m)$ is m-dimensional. A reader who is not familiar with differential geometry will not lose very much by assuming that Q is an open subset of some $\mathbb{R}^d$. Write

$$A_j(q) = \sum_{k=1}^{d} \theta_j^k(q) \frac{\partial}{\partial q^k} ,$$

where $q = (q^1, \ldots, q^d)$, then the first line of (1) is equivalent to

$$\dot{q}^k(t) = \theta_0^k(q) + \sum_{i=1}^{m} u_i(t) \theta_i^k(q), \qquad k = 1, \ldots, d.$$

The dimension of the control-vector is greater than or equal to the dimension of the output-vector, i.e., $m \geq \ell$

We are considering static state-feedbacks:

$$u = \alpha(q) + \beta(q) v$$

where

- $v = (v^1, \ldots, v^m)$ is the new control,
- $\alpha(q)$ (resp. $\beta(q)$) is a m-vector (resp. a m×m invertible matrix) which is state-dependent.

We want the k-th component y^k of the output to be influenced only by the k-th component v^k of the new control-vector. The characteristic number ρ_k of y^k is defined by: $\exists\, i \in \{1, \ldots, m\}$ such that $A_i (A_0)^{\rho_k} h^k(q) \not\equiv 0$; $\forall \nu$, $0 \leq \nu < \rho_k$, $\forall i \in \{1, \ldots, m\}$, $A_i (A_0)^{\nu} h^k(q) \equiv 0$.

Remark. The characteristic number may not be defined, i.e., infinite. This means that y^k is not affected by the control-vector u. If the characteristic number is defined, Claude [2] has given an upper bounds of it that is related to the dimention of Q.

Define the characteristic matrix C(q) by

$$C(q) = [A_i (A_o)^{\rho_k} h^k (q)]_{\substack{i=1,\dots,m \\ k=1,\dots,\ell}} .$$

Theorem 1:(1) The noninteracting control problem for system (1) is solvable by static state-feedback if, and only if, the characteristic matrix has rank ℓ.

The meaning of this condition is the following one: The system of linear equations

$$C(q)u = \frac{d^{\rho_k+1} y}{dt^{\rho_k+1}} - (A_o)^{\rho_k+1} h(q)$$

is solvable with respect to u iff the rank condition is satisfied.

B. Towards a Differential Algebraic Viewpoint

1) System definition

Consider the following system

$$\begin{cases} \dot{q} = F(q,u) \\ \\ y = h(q) \ . \end{cases} \tag{2}$$

The state $q = (q^1,\dots,q^d)$, the control $u = (u^1,\dots,u^m)$, the output $y = (y^1,\dots,y^\ell)$ are vectors of various dimensions, $\ell \le m$. In order to employ differential algebra, F and h must satisfy some restrictive conditions and so, F and h cannot be completely arbitrary (cf. Johnson [12]). There is no problem if $F = (F^1,\dots,F^d)$, $h = (h^1,\dots,h^\ell)$(2) are rational functions of $u^1,\dots,u^m$, $q^1,\dots,q^d$. When dealing with more general functions we will have to introduce some classical terminology and notations (cf. Kaplansky [13](3), Kolchin [14,15], Pommaret [18]).

A derivation of a ring R is an additive mapping $a \mapsto \dot{a}$ of R into itself such that

$$(ab)^{\cdot} = \dot{a}b + a\dot{b} .$$

---

(1) For a very interesting and early paper related to that result see Asseo [1].

(2) It is straightforward to extend our results to the case where the output map h is also control dependent, i.e., $y = h(q,u)$.

(3) Kaplansky's book is a marvelous introduction to the subject. Unfortunately it does not cover enough material for our purpose.

A differential ring is a commutative ring with a derivation (here the time-derivation). Introduce the differential field $K := \mathbb{R}\langle t\rangle$; elements of K are rational functions of t with real coefficients. We want to be able to employ algebraic functions like $\sqrt{1+(q^1)^2}$ or elementary transcendential functions like $\sin u^1$. They are all solutions of algebraic differential equations. This is translated into the following fundamental assumption that enables us to treat any realistic case-study:

The differential field $K\langle u,q,F,h\rangle$ is a differential algebraic extension of the differential field $K\langle u,q\rangle$.

2) Feedback laws

We will be considering here feedback laws that are more general than the purely static ones. We will be employing elementary differential Galois theory.

A feedback law is a differential isomorphism $\varphi: K\langle u\rangle \to K\langle v\rangle$, $u^i \mapsto v^i$, $i=1,\dots,m$, which leaves K elementwise fixed (a differential isomorphism is an isomorphism which commutes with the derivation, i.e.,

$$\varphi(\dot{u}^i) = \frac{d}{dt}\varphi(u^i).$$

A differential ideal (resp. a prime differential ideal) of a differential ring is an ideal (resp. a prime ideal) like in "ordinary" commutative algebra, which is also a differential subring. Let P be the prime differential ideal in the differential ring $K\{u,q,F\}$ generated by the state-equations $\dot{q} - F(q,u) = 0$ of system (2). Let L be the quotient field of $K\{u,q,F\}/P$. A differential rational static state-feedback is defined in the following way:

There exists a differential algebraic extension $L\langle\eta\rangle$, $\eta = (\eta^1,\dots,\eta^s)$, of L such that $K\langle v\rangle \subset L\langle\eta\rangle$. Then $v^i = \phi^i(q,F,\dots,F^{(\alpha)},\eta,\dots,\eta^{(\beta)}, u,\dots,u^{(j)})$ $\left(f^{(\nu)} = \frac{d^\nu f}{dt^\nu}\right)$, where ϕ^i is a rational function in all of its variables. Set $\bar{u} = u^{(j)}$, $\dot{\xi}^1 = \bar{u},\dots,\dot{\xi}^{\alpha-1} = \xi^\alpha$. We get $v = \phi(q,F,\dots,F^{(\alpha)}, \eta,\dots,\eta^{(\beta)},\xi^\alpha,\dots,\xi^1,\bar{u})$. That was the type of dynamic feedback already used in [3,8,9,24].

We are thus employing only differentiations and rational computations. As stated in the introduction this fact is a property of all the known case-studies with feedback or extended feedback. For purely static feedback those computations received a most interesting graph theoretic interpretation (Géromel, Lévine, Willis [6]) which seems to be generalizable for the differential case [16].

3) The noninteracting control problem

We want to find feedback laws $u \mapsto v = \varphi(u)$ such that the component y^i of the output is influenced only by the component v^i of the feedback modified control. For linear systems, the problem is solvable with dynamic compensation iff the system is right-invertible, i.e., if its transfer matrix is right-invertible (cf. Wolovich [26]). Before stating our result we must summarize our approach to nonlinear right-invertibility [4].

Roughly speaking system (2) is said to be right-invertible iff the components $y^1,\ldots,y^\ell$ of the outputs do not satisfy any differential equation which is independent of the control u and the state q. We need to translate this into a more mathematical language. In the differential Zariski topology, closed sets are the sets of functions which satisfy some given differential equations. System (2) is said to be dominant iff the closure with respect to the differential Zariski topology of all the possible output functions gives in some sense all possible functions. It is known (cf. Silverman and Payne [23]) that right-invertible linear systems are not surjective. It can be shown that they are characterized by being dominant. The differential transcedence degree$^{(4)}$ of the differential field generated by $h^1(q),\ldots,h^\ell(q)$, a most important invariant integer, is equal to ℓ iff system (2) is dominant. For linear systems it turns out that this degree is the rank of the transfer matrix.

It should be clear now for the reader that our main result is a complete generalization of the linear theory:

Theorem 2: The noninteracting control problem for system (2) is solvable if, and only if, system (2) is dominant. Then a differential static state-feedback can be computed such that the new system exhibits a linear, decoupled input-output behavior.

It is not appropriate to reproduce the complete proof here; it will be given elsewhere. It employs a generalization of the so-called Kahler differentials to a differential algebraic setting (Johnson [12]). This permits a kind of "linearization" of the computations involved in the elimination process. Our approach should be considered as a nonlinear analogue of Silverman's proof [22] of the decoupling of linear systems by using the structure algorithm. We end up with the linear equations

$$\frac{d^{\alpha_k} y^k}{dt^{\alpha_k}} = v^k .$$

4) An easy example [4]

We want to solve the noninteracting control problem for the following bilinear system

$$\begin{cases} \dot{q}^o = u^2 \\ \dot{q}^1 = u^1 q^o \\ \dot{q}^2 = u^1 q^1 \\ y^1 = q^1 \\ y^2 = q^2 . \end{cases}$$

Purely static feedback does not work since the rank of this system's characteristic matrix is one. We leave it to the reader to verify that the following dynamic compensation is satisfactory

---

(4) The concept of differential transcendence degree extends the classical transcendence degree in field theory.

$$\begin{cases} \dot{\xi} = u^2 \\ v^1 = u^1 q^o q^1 \xi \\ v^2 = u^1 q^1 . \end{cases}$$

ACKNOWLEDGMENT

The author would like to thank Daniel Connors for revising the English of this paper.

REFERENCES

[1] Asseo, S. J., "Decoupling of a Class of Nonlinear Systems and its Application to an Aircraft Control Problem," J. Aircraft, 10, 1973, pp. 739-747.

[2] Claude, D., "Découplage des systèmes non linéaires, séries génératrices non commutatives et algèbres de Lie," SIAM J. Control Optimization, to appear.

[3] Descusse, J. and C. H. Moog, "Decoupling with Dynamic Compensation for Strong Invertible Affine Nonlinear Systems," Int. J. Control, to appear.

[4] Fliess, M., "Some Remarks on Nonlinear Invertibility and Dynamic State-Feedback," MTNS-85, Stockholm, June 1985, C. Byrnes and A. Lindquist, eds., Elsevier, Amsterdam, to appear.

[5] Freund, E., "The Structure of Decoupled Nonlinear Systems," Int. J. Control, 21, 1975, pp. 651-659.

[6] Géromel, F., J. Lévine, and P. Willis, "A Fast Algorithm for Systems Decoupling Using Formal Calculus," in Analysis and Optimization, Part 2, A. Bensoussan and J. L. Lions, eds., Lecture Notes in Control and Information Sciences, 63, Springer, Berlin, 1984, pp. 378-390.

[7] Hirschorn, R. M., "(A,B)-Invariant Distributions and Disturbance Decoupling of Nonlinear Control Systems," SIAM J. Control and Optimization, 19, 1981, pp. 1-19.

[8] Isidori, A., Nonlinear Control Systems: An Introduction, Lecture Notes in Control and Information Sciences, 72, Springer, Berlin, 1985.

[9] Isidori, A., "Control of Nonlinear Systems via Dynamic State-Feedback," Actes Conf. Methodes Algebriques Geometriques Automatique non Lineaire, Paris, June 1985, M. Fliess and M. Hazewinkel, eds., D. Reidel, Dordrecht, to appear.

[10] Isidori, A., A. J. Krener, C. Gori-Giorgi, and S. Monaco, "Nonlinear Decoupling via Feedback: A Differential Geometric Approach," IEEE Trans. on Automatic Control, Vol. AC-26, 1981, pp. 331-345.

[11] Johnson, J., "Kahler Differentials and Differential Algebra," Annals of Math., 89, 1969, pp. 92-98.

[12] Johnson, J., "Orders for Systems of Differential Equations and a Generalization of the Notion of Differential Ring," J. Algebra, 78, 1982, pp. 91-119.

[13] Kaplansky, I., An Introduction to Differential Algebra, 2nd Ed., Hermann, Paris, 1976.

[14] Kolchin, E. R., Differential Algebra and Algebraic Groups, Academic Press, New York, 1973.

[15] Kolchin, E. R., Differential Algebraic Groups, Academic Press, New York, 1985.

[16] De Luca, A., A. Isidori, and F. Nicolo, "An Application of Nonlinear Model Matching to the Dynamic Control of Robot Arm with Elastic Joints," Proc. IFAC Symp. Robot Control, Barcelona, November 1985.

[17] Nijmeijer, H. and A. J. van der Schaft, "Controlled Invariance for Nonlinear Systems," IEEE Trans. on Automatic Control, AC-27, 1982, pp. 904-914.

[18] Pommaret, J. F., Differential Galois Theory, Gordon and Breach, New York, 1983.

[19] Porter, W. A., "Diagonalizations and Inverses for Nonlinear Systems," Int. J. Control, 11, 1970, pp. 67-76.

[20] Ritt, J. F., Differential Algebra, American Math. Soc., Providence, RI, 1950.

[21] van der Schaft, A. J., "Linearization and Input-Output Decoupling for General Nonlinear Systems," Systems and Control Letters, 5, 1985, pp. 27-33.

[22] Silverman, L. M., "Decoupling with State Feedback and Precompensation," IEEE Trans. on Automatic Control, AC-15, 1970, pp. 487-489.

[23] Silverman, L. M. and H. J. Payne, "Input-Output Structure of Linear Systems with Applications to the Decoupling Problem," SIAM J. Control, 9, 1971, pp. 199-233.

[24] Singh. S. N., "Generalized Decoupled-Control Synthesis for Invertible Nonlinear Systems," IEE Proc., Pt. D, 128, 1981, pp. 157-161.

[25] Sinha, P. K., "State Feedback Decoupling of Nonlinear Systems," IEEE Trans. on Automatic Control, AC-22, 1977, pp. 487-489.

[26] Wolovich, W. A., Linear Multivariable Systems, Springer, New York, 1974.

[27] Wonham, W. M., Linear Multivariable Control: A Geometric Approach, 3rd Ed., Springer, New York, 1985.

ADAPTIVE HIGH-GAIN FEEDBACK CONTROL

A. SABERI
Dept. of Electrical
& Computer Eng.
Washington State Univ.
Pullman, WA 99164

H. KHALIL
Dept. of Electrical Eng.
& Systems Science
Michigan State University
East Lansing, MI 48824

ABSTRACT

The paper presents a high-gain feedback stabilizing control method in which the high-gain parameters are adapted on line. The method is applied to uniform rank multivariable linear systems as well as a class of nonlinear systems. The system can be unknown except for the information that is usually required in the adaptive control literature.

INTRODUCTION

It is well-known that high-gain feedback can be used to achieve arbitrary desensitization and stabilization of feedback control systems in which the plant is minimum phase. Multivariable results using different techniques are available in the literature, e.g., [1 - 6]. It has been emphasized recently [4], [5] that high-gain feedback systems also possess a degree of robustness that results from the fact that high-gain feedback designs are dependent mainly on the high-frequency behavior of the plant. For example, a single-input single-output plant can be stabilized by high-gain feedback if we know its relative degree, the sign of the high-frequency gain and that all its zeros are in the open left-half plane. These three pieces of information in addition to knowing an upper bound on the order of the plant is what is needed to adaptively stabilize the plant [7], [8] . In the case of high-gain feedback one needs additional information in order to actually design a high-gain compensator. Using the above three pieces of information one can determine the structure of the compensator and its parameters except for a number of scalar parameters which are required to be either sufficiently large or sufficiently small. The choice of these parameters requires additional information on the plant, e.g., bounds on plant parameters, bounds on plant poles, etc. One idea to alleviate the need to know this additional information is to adaptively tune these parameters on line. For example, a high-gain parameter K can be tuned using the adaptation law $\dot{K} = ||y(t)||^2$ so that K(t), which is monotonically nondecreasing, will increase until it becomes large enough to force the asymptotic stability of the closed-loop system. Then $||y(t)||$ will be convering to zero and K(t) will be converging to a limiting value. The idea of adaptively tuning the high-gain parameter has been used before by Willems and Byrnes [9], as a first step towards their adaptive stabilization scheme which employs a switching strategy to stabilize a plant in the absence of information on the sign of its high frequency gain. If the sign of the high frequency gain is known, their switching strategy is not needed and their scheme reduces to the high-gain adaptation idea we described above. The results of Willems and Byrnes [9], however, are given only for relative degree one systems where the high-gain compensator can be taken as constant output feedback. In systems with relative degree higher than one, the situation is more complicated because the high-gain compensator must include poles and/or zeros which asymptotically tend to infinity as the high-gain parameter tends to infinity. Root locus arguments, like the one used in [9], cannot easily handle this case.

Section 2 of this paper presents an adaptive high-gain feedback control method for the stabilization of systems with relative degree higher than one. Using information about the infinite frequency behavior of a minimum phase system, a high-gain compensator is designed with all parameters fixed except for a number of scalar parameters which are required to be sufficiently small. These parameters are monotonically reduced on line using an adaption law. To show convergence and asymptotic stability we construct a Lyapunov function candidate for the closed-loop system. The function is not dependent on the adapted parameters but becomes a Lyapunov function once these parameters become sufficiently small. The Lyapunov function is used then to show that the trajectory of the closed-loop system converges to the origin and that the adapted parameters converge to limiting values. The construction of the Lyapunov function is done in two steps. First, a singularly perturbed state-space realization of the closed-loop system is obtained. Second, the Lyapunov analysis of Saberi and Khalil [10] is employed.

The results of Section 2 can be worked out for any square, invertible minimum phase plant and for various high-gain compensator design techniques. In order to present the idea and the Lyapunov analysis in a simple form we chose to restrict our analysis to a class of invertible plants for which the lowest-order nonzero Markov parameter is nonsingular (i.e., uniform rank plants). We also chose a high-gain compensator design similar to the one used by Zames and Bensoussan [5] in order to simplify the time scale structure of the closed-loop system and subsequently simplify the singularly perturbed equations. We emphasize, however, that the same analysis can be applied to high-gain compensators like, for example, the observer based compensator used by Kwakernaak [4].

The use of singular perturbations and the Lyapunov analysis of [10] allows nonlinear generalization of the results in Section 2. In Section 3 we consider a class of nonlinear systems which amounts to a generalization of uniform rank linear systems. We give conditions under which the same adaptive high-gain feedback controller used in Section 2 will stabilize the nonlinear system.

2. LINEAR MULTIVARIABLE SYSTEMS

We consider an m x m multivariable linear plant of order n described by a strictly proper transfer function P(s). We assume

(i) $P(s) = \sum_{i=q}^{\infty} M_i s^{-i}$, $q \geq 1$, M_q is nonsingular,

(ii) All the zeros of P(s) are in the open left-half complex plane,

(iii) The integer q is known,

and

(iv) An m x m matrix M, which satisfies the eigenvalue condition $R_e\lambda(M_qM) > 0$, is known.

For a single-input single-output plant (m = 1), assumption (i) is automatically satisfied, q is the relative degree of the system which is the difference between the number of poles and the number of zeros, and the high-frequency condition (iv) reduces to knowing the sign of the high-frequency gain. For a plant satisfying (i), Kwakernaak's method[1] [4] requires assumptions (ii) - (iv), in addition to knowing the plant order n. Thus for the class of systems satisfying (i), our assumptions (ii) - (iv) are slightly weaker than those used in [4].

A high-gain compensator C(s) in a conventional feedback configuration is taken as

$$C(s) = \frac{1}{\varepsilon} MN(s) \frac{1}{(\mu s+1)^{q-1}} \tag{1}$$

where

$$N(s) = I_m s^{q-1} + N_{q-1} s^{q-2} + . . . + N_1 \tag{2}$$

and the matrices $N_1, . . ., N_{q-1}$ are chosen such that the zeros of det N(s) = 0 are in the open left-half complex plane. The calculation of $N_1, . . ., N_{q-1}$ can be represented as a state-feedback pole placement problem. The scalar parameters ε and μ are positive constants which will be chosen to satisfy

$$\varepsilon << 1 \text{ and } \mu/\varepsilon << 1 \tag{3}$$

The compensator (1) is very similar to the compensator used by Zames and Bensoussan [5] in that it has a polynomial part N(s) and a strictly proper part $1/(\mu s+1)^{q-1}$. In [5] the polynomial part inverts the plant while the strictly proper part ensures the strict properness of the compensator. In our compensator (1), the polynomial part and the plant form a relative degree one system while the strictly proper part $1/(\mu s+1)^{q-1}$ ensures the properness of the compensator C(s). These deviations from Zames and Bensoussan's compensator were adopted so that in the case q = 1 the compensator C(s) reduces to the constant gain form

$$C = \frac{1}{\varepsilon} M \tag{4}$$

which was used in Byrnes and Willems [9] as well as in other studies of high-gain feedback for the case q = 1 (e.g. [1]).

In view of assumption (i), a minimal state-space realization of P(s) is given by [6]

$$\dot{x}_0 = A_{00}x_0 + A_{01}x_1 \tag{5}$$

$$\dot{x}_i = x_{i+1}, \quad i = 1, 2, . . ., q - 1 \tag{6}$$

$$\dot{x}_q = M_q(u + \sum_{i=0}^{q} D_i x_i) \tag{7}$$

$$y = x_1 \tag{8}$$

---

[1] Kwakernaak's method is not limited to plants satisfying assumption (i). It works with invertible plants.

where the eigenvalues of A_{00} are the zeros of $P(s)$; hence by assumption (ii) they are in the open left-half complex plane. With the realization (5) - (8) and the compensator $C(s)$ defined in (1), a state-space representation of the closed-loop system is given by

$$\dot{x}_0 = A_{00}x_0 + A_{01}x_1 \tag{9}$$

$$\dot{x}_i = x_{i+1}, \quad i = 1, \ldots, q - 1 \tag{10}$$

$$\varepsilon\dot{x}_q = \varepsilon M_q \sum_{i=0}^{q} D_i x_i - M_q M z_1 \tag{11}$$

$$\mu\dot{z}_j = -z_j + z_{j+1}, \quad j = 1, \ldots, q - 2 \tag{12}$$

$$\mu\dot{z}_{q-1} = -z_{q-1} + \sum_{i=1}^{q-1} N_i x_i + x_q \tag{13}$$

For $q = 1$, equations (12) and (13) drop out and z_1 in (11) is replaced by x_1. For $q = 2$, equation (12) drops out. When ε and $\frac{\mu}{\varepsilon}$ are small, the system (9) - (13) is a three-time-scale singularly perturbed system. Defining $x_s = (x_0^T, x_1^T, \ldots, x_{q-1}^T)^T$ and $z = (z_1^T, z_2^T, \ldots, z_{q-1}^T)^T$, we can rewrite (9) - (13) in the compact form

$$\dot{x}_s = F_{11}x_s + F_{12}x_q \tag{14}$$

$$\varepsilon\dot{x}_q = \varepsilon F_{21}x_s + \varepsilon F_{22}x_q + F_{23}z \tag{15}$$

$$\mu\dot{z} = F_{31}x_s + F_{32}x_q + F_{33}z \tag{16}$$

From singular pertubation theory [11] we know that as $\varepsilon \to 0$ and $\mu/\varepsilon \to 0$, the eigenvalues of (14) - (16) approach the eignevalues of A_1, A_2/ε and A_3/μ where

$$A_3 = F_{33} = \begin{bmatrix} -I & I & & & & 0 \\ & -I & I & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & \cdot & I \\ 0 & & & & & -I \end{bmatrix} \tag{17}$$

$$A_2 = -F_{23}F_{33}^{-1}F_{32} = -M_qM \tag{18}$$

and

$$A_1 = F_{11} - F_{12}(F_{23}F_{33}^{-1}F_{32})^{-1}(F_{23}F_{33}^{-1}F_{31}) = \left[\begin{array}{c|ccc} A_{00} & A_{01} & 0 & 0 \\ \hline 0 & 0 & I & \\ \cdot & & \ddots & 0 \\ \cdot & & & \ddots \\ \cdot & & & I \\ 0 & -N_1 & -N_2 & -N_{q-1} \end{array}\right] \tag{19}$$

which shows the asymptotic stability of the system for sufficiently small ε and μ/ε. To construct a Lyapunov function for the closed-loop system (14) - (16), we employ the Lyapunov stability analysis of singularly perturbed systems from Saberi and Khalil [10]. Let P_1, P_2 and P_3 be the unique positive definite solutions of the algebraic Lyapunov equation

$$P_iA_i + A_i^TP_i = -I_i, \quad i = 1, 2, 3. \tag{20}$$

A Lyapunov function candidate for (14) - (16) is taken as

$$V(x_s, x_q, z) = x_s^TP_1x_s + x_f^TP_2x_f + z_f^TP_3z_f \tag{21}$$

where

$$x_f = x_q + (F_{23}F_{33}^{-1}F_{32})^{-1}(F_{23}F_{33}^{-1}F_{31})x_s = x_q + \sum_{i=1}^{q-1} N_ix_i$$

$$z_f = z + F_{33}^{-1}F_{31}x_s + F_{33}^{-1}F_{32}x_q = [(z_1 - x_f)^T, \ldots, (z_{q-1} - x_f)^T]^T$$

It can be shown that for any ε and μ the derivative of V along the trajectory of (14) - (16) satisfies

$$\begin{aligned}\dot{V} \le &- \|x_s\|^2 + c_1\|x_s\|\,\|x_f\| + c_2\|x_s\|\,\|z_f\| - (\tfrac{1}{\varepsilon} - c_3)\|x_f\|^2 \\ &+ (\tfrac{c_4}{\varepsilon} + c_5)\|x_f\|\,\|z_f\| - (\tfrac{1}{\mu} - \tfrac{c_6}{\varepsilon})\|z_f\|^2\end{aligned} \tag{22}$$

where the nonnegative constants c_i depend on the norms of the matrices F_{ij} and P_i but they are independent of ε and μ. From (22) we obtain

$$\dot{V} \le - \phi^T T\phi - 2\alpha V \tag{23}$$

where

$$\phi^T = [\|x_s\|, \|x_f\|, \|z_f\|],$$

the symmetric matrix T is given by

$$T = \begin{bmatrix} 1-2\alpha\lambda_1 & -\frac{1}{2} c_1 & -\frac{1}{2} c_2 \\ & \frac{1}{\varepsilon} - c_3 - 2\alpha\lambda_2 & -\frac{1}{2} (\frac{c_4}{\varepsilon} + c_5) \\ & & (\frac{1}{\mu} - \frac{c_6}{\varepsilon} - 2\alpha\lambda_3) \end{bmatrix} \qquad (24)$$

and $\lambda_i = \lambda_{max} (P_i)$. Choosing $\alpha = 1/4 \lambda_1$, it can be easily seen from (24) that for sufficiently small ε and μ/ε the matrix T is positive definite. Then

$$\dot{V} \leq -2 \alpha V \qquad (25)$$

and the closed-loop system is exponentially stable. If the constants c_i are known one can actually calculate bounds on ε and μ/ε such that T is positive definite for all ε and μ/ε less than these bounds.

To implement the compensator c(s) we need to choose ε and μ/ε small enough. The smallness of μ/ε can be ensured by taking

$$\mu = \varepsilon^{\beta}, \qquad \beta > 1 \qquad (26)$$

so that $\mu/\varepsilon \to 0$ as $\varepsilon \to 0$. Implementing c(s) with a constant (sufficiently small) value for ε requires additional information about the plant over what is required in assumptions (i) - (iv). We can instead decrease ε on line until it becomes small enough. In [9], ε was taken as $\varepsilon = 1/K$ where $\dot{k} = || y ||^2$. Using an adaption law like this in our case will cause a problem in realizing the network

$$(\mu D + 1)^{q-1} z_1(t) = N(D)y(t), \qquad D = \frac{d}{dt} \qquad (27)$$

since μ will be a time-varying parameter and the realization will depend on derivatives of μ which are not available on line. Therefore we decided to decrement ε in a step-wise fashion using the adaption law

$$\dot{K} = \gamma || y(t) ||^{2}, \qquad \gamma > 0, K(t_o) > 0 \qquad (28)$$

$$\varepsilon(t) = 1/K(t_i), \qquad t_i \leq t < t_{i+1} \qquad (29)$$

where the switching points t_i are chosen to satisfy

$$K(t_i) = K(t_o) + ai, \qquad a > 0 \qquad (30)$$

Our adaptive control algorithm is now completely defined by (26) - (30) and

$$u(t) = - \frac{M}{\varepsilon(t)} z_1(t) \qquad (31)$$

Theorem 1: Suppose that the plant satisfies assumptions (i) to (iv) and the adaptive control algorithm is given by (26) - (31). Then for all initial conditions of the plant, the network (27) and the adaption law (28),

$$K(t) \to K_\infty < \infty \quad \text{as } t \to \infty \tag{32}$$

and the state response $X = (x_s^T, x_q^T, z^T)^T$ is bounded and satisfies

$$X(t) \to 0 \quad \text{as} \to t \tag{33}$$

Proof: The key point in the proof is to show that K(t) is bounded and hence there is a finite number of switching points t_i. Suppose, to the contrary, that $i \to \infty$. Over any time period $[t_i, t_{i+1})$, the Lyapunov analysis outlined before is applicable. Since the function V, defined by (21), is independent of ε and μ, we see that for sufficiently large i, K will be large enough so that (25) holds uniformly in i. Thus there exists a finite integer i* such that

$$\| y(t) \| \le K_1 e^{-\alpha(t-T)} \| y(T) \|, \qquad t \ge T \ge t_{i^*} \tag{34}$$

where K_1 and α are independent of i. It follows that

$$\int_{t_i^*}^{\infty} \| y(t) \|^2 dt \le \frac{K_1^2}{2\alpha} \| y(t_{i^*}) \|^2 < \infty$$

Hence K(t) is bounded, and from (30) the number of switching points is finite. The fact that K(t) is also monotonically nondecreasing shows (32). Moreover, (28) implies that $y(t) \to 0$ as $t \to \infty$ which, from the closed-loop equations (9) - (13), implies (33).

3. A CLASS OF NONLINEAR SYSTEMS

We consider a nonlinear system described by

$$\dot{x}_0 = f_0(x_0, x_1) \tag{35}$$

$$\dot{x}_i = x_{i+1}, \qquad i = 1, \ldots, q-1 \tag{36}$$

$$\dot{x}_q = f_q(x_0, \ldots, x_q) + M_q(x_0, \ldots, x_{q-1})u \tag{37}$$

$$y = x_1, \qquad u, y \in R^m, \; x \in R^n \tag{38}$$

which can be viewed as a generalization of the linear system (5)-(8). Following [12] the equation

$$\dot{x}_0 = f_0(x_0, 0) \tag{39}$$

will be referred to as the zero dynamics (i.e., the dynamics when y is restricted to zero). The following assumptions are supposed to hold globally for all $x \in R^n$:

(a) x = 0 is an equilibrium point of the system (35)-(37) when u = 0, and it is the unique equilibrium point when y = 0 as well.

(b) The zero dynamics $\dot{x}_0 = f_0(x_0, 0)$ is exponentially stable.

(c) There exists a matrix M and a symmetric positive-definite matrix P_2 such that

$$P_2 M_q(x_0, \ldots, x_{q-1}) M + M^T M_q^T(x_0, \ldots, x_{q-1}) P_2 = I \tag{40}$$

(d) The functions f_0 and f_q satisfy global Lipschitz conditions and M_q is bounded.

Assumption (a) allows the system to have open-loop multiple equilibra as long as y vanishes only at the origin. Assumption (c) holds whenever $M_q(x_0, \ldots, x_{q-1})$ is positive-definite uniformly in $x_0, \ldots, x_{q-1}$. In such a case M and P_2 can be identity matrices. In single input-single output systems assumption (c) reduces to requiring $M(x_0, \ldots, x_{q-1})$ to have a definite sign which is known. Under assumptions (a)-(d), the system (35) - (38) is similar to the linear system studied in Section 2 when it comes to adaptive high-gain feedback stabilization. Specifically, it can be stabilized by the same adaptive control algorithm (26) - (31) as it is demonstrated by the following theorem.

Theorem 2. Suppose that the nonlinear system (35) - (38) satisfies assumptions (a) - (d), and the adaptive control algorithm is given by (26) - (31). Then for all initial conditions of the system (35) - (38), the network (27) and the adaptation law (28),

$$K(t) \to K_\infty < \infty \qquad \text{as } t \to \infty \tag{41}$$

and

$$X(t) \to 0 \qquad \text{as } t \to \infty \tag{42}$$

where $X(t) \in R^{n+m(q-1)}$ is the state of the closed-loop system.

Proof: Consider any period $[t_i, t_{i+1})$. A state-space realization of the closed-loop system is given by

$$\dot{x}_0 = f_0(x_0, x_1) \tag{43}$$

$$\dot{x}_i = x_{i+1}, \qquad i = 1, \ldots, q-1 \tag{44}$$

$$\varepsilon \dot{x}_q = \varepsilon f_q(x_0, \ldots, x_q) - M_q(x_0, \ldots, x_{q-1}) M z_1 \tag{45}$$

$$\mu \dot{z}_j = -z_j + z_{j+1}, \qquad j = 1, \ldots, q-2 \tag{46}$$

$$\mu \dot{z}_{q-1} = -z_{q-1} + \sum_{i=1}^{q-1} N_i x_i + x_q \tag{47}$$

The reduced equations of this singularly perturbed system, obtained by setting $\varepsilon = \mu = 0$, are given by

$$\dot{x}_0 = f_0(x_0, x_1) \tag{48}$$

$$\dot{\bar{x}}_s = A_0 \bar{x}_s \tag{49}$$

where $\bar{x}_s = (x_1^T, x_2^T, \ldots, x_{q-1}^T)^T$ and $\operatorname{Re} \lambda(A_0) < 0$.

Using exponential stability of the zero dynamics we can show that the reduced system has a Lyapunov function V_1 such that

$$\dot{V}_1 \leq -\alpha_1 V_1, \qquad \alpha_1 > 0 \tag{50}$$

which shows that the reduced system (48), (49) is exponentially stable. Now a Lyapunov function candidate for the closed-loop system (43) - (47) is taken as

$$V = V_1(x_s) + x_f^T P_2 x_f + z_f^T P_3 z_f$$

where $P_3 > 0$ satisfies (20) and x_f and z_f are defined after (21). From this point on the proof of Theorem 2 proceeds exactly as that of Theorem 1, where we make use of the fact that assumption (d) guarantees that there will be no finite escape times, until we conclude that K(t) is bounded and y(t) is square integrable. Using a detailed argument, which is omitted here, it can be shown that the square integrability of y ensures that the state of the closed-loop system converges to 0.

Acknowledgement: The work of H. Khalil was supported by the National Science Foundation under grant no. ECS-8410649.

REFERENCES

1. Young, K.K.D., P.V. Kokotovic and V.I. Utkin, "A singular perturbation analysis of high-gain feedback systems," IEEE Trans. on Automatic Control, Vol. AC-22, pp. 931-939, 1977.

2. Kimura, H., "Perfect and subperfect regulation in linear multivariable control systems, Automatica, Vol. 17, 1981.

3. Willems, I.C., "Almost invariant subspaces: an approach to high-gain feedback design, Part II," IEEE Trans. on Automatic Control, Vol. Ac-27, pp. 1071-1085, 1982.

4. Kwakernaak, H., "A condition for robust stabilizability," Systems and Control Letters, Vol. 2, pp. 1-5, 1982.

5. Zames, G. and D. Bensoussan, "Multivariable feedback, sensitivity and decentrailized control," IEEE Transactions on Automatic Control, Vol. AC-2g, pp. 542-550, 1984.

6. Sannuti, P., "Direct singular perturbation analysis of high-gain and cheap control problems," Automatica, Vol. 19, pp. 41-51, 1983.

7. Morse, A.S., "Global stability of parameter-adaptive control systems," IEEE Trans. on Automatic Control, Vol. AC-25, pp. 433-439, 1980.

8. Narendra, K.S., Y.H. Lin and L.S. Valavani, "Stable adaptive controller design, Part II: Proof of stability," IEEE Trans. on Automatic Control, Vol. Ac-25, pp. 440-448, 1980.

9. Willems, J.C. and C.I. Byrnes, "Global adaptive stabilization in the absence of information on the sign of the high frequency gain," Proc. INRIA Conf. on Analysis and Optimization of Systems, pp. 41-57, Springer-Verlag, 1984.

10. Saberi, A. and H.K. Khalil, "Quadratic-type Lyapunov functions for singularly perturbed systems," IEEE Trans on Automatic Control, Vol. AC-2g, pp. 542-550, 1984.

11. Kokotovic, P.V., H.K. Khalil and J. O'Reilly, "Singular Perturbation Methods in Control: Analysis and Design, Academic Press, 1986.

12. Byrnes, C.I. and A. Isidori, "A frequency domain philosophy for nonlinear systems, with application to stabilization and to adaptive control," Proc. of 23rd CDC Conf., Las Vegas, NV, pp. 1569-1573, 1984.

A LYAPUNOV APPROACH FOR THE DESIGN OF VARIABLE STRUCTURE SYSTEMS

HEBERTT SIRA-RAMIREZ
Control Systems Department. Universidad de Los Andes.
Merida-Venezuela.

Abstract This article presents a design method for the sliding mode creation in VSS [1] using a Lyapunov function approach.

Consider a linear dynamic system $\dot{\underline{x}} = A\underline{x}+\underline{b}u$ with $u = -\underline{f}^T\underline{x}$ and $(A,\underline{b})$ controllable. Let $V(\underline{x}) = \underline{x}^TP\underline{x}$ be a Lyapunov function candidate. Krasovski's sinthesis technique [2] defines a Variable Structure Controller (VSC) of the form $f_i = \beta_i$ for $sx_i < 0$ and $f_i = \alpha_i$ for $sx_i > 0$ where $\alpha_i > 0$ and $\beta_i < 0$ $(i=1,2,...,k)$ are the bounds for the feedback gains and $s = \underline{b}^TP\underline{x} = 0$ is the switching surface. P is obtaied as a positive solution of $A^TP+PA=-Q$. This "modulated bang-bang" feedback control minimizes a standard quadratic performance index defined on the state trajectories alone (no control penalty) and results in an asymptotically stable closed loop system.

A sliding regime can be created on $s = 0$ if and only if one prescribes appropriate values for the bounds of the feedback gains f_i, which yield $s\, ds/dt < 0$. These are $\beta_i \leq m_i$; $\alpha_i \geq m_i$ with $m_i = [(\underline{b}^TPA)_i-(\underline{b}^TPA)_n(\underline{b}^TP)_i/(\underline{b}^TP)_n]/(\underline{b}^TP\underline{b})$; $i=1,2,...,k$. The number of feedback loops k is determined from the fact that only $n-k-1$ equalities of the form: $(\underline{b}^TPA)_i/(\underline{b}^TP)_i = (\underline{b}^TPA)_n/(\underline{b}^TP)_n$ are to be satisfied.

The active switching of the VSC keeps the state on $s=0$ in a sort of jittering motion that slides towards its stable equilibrium. Ideally this motion satisfies $s=0$ and $ds/dt=0$ leading to the "equivalent control problem". This control would keep the state trajectory on $s=0$ if no perturbations were present. Under these circumstances, the ideal sliding dynamics is given by $d\underline{x}/dt = (A-\underline{b}\underline{b}^TPA/\underline{b}^TP\underline{b})\underline{x}$ on $s=0$. Except for a zero at the origin these eigenvalues coincide with the transmission zeros of the open loop system with output $y = \underline{b}^TP\underline{x}$. This establishes an affine relationship among the coefficients of the open loop characteristic polynomial and those of the ideal sliding. The setting of the latter yields the former which in turn are achievable by linear feedback on the original plant.

Consider the identity $V(\underline{x}) = \underline{x}^TP\underline{x} = \underline{x}^T(P-P\underline{b}\underline{b}^TP/\underline{b}^TP\underline{b})\underline{x}+s^2/\underline{b}^TP\underline{b}$. All of the ingredients for the sliding regime formation are present in this *ideal sliding dynamics based decomposition* of $V(\underline{x})$. Under a suitable change of coordinate $\underline{z} = P_1\underline{x}$ with $P_1{}^TP_1 = P$ this decomposition is equivalent to the *geometrically based decomposition* of $\underline{x}^T\underline{x}$ obtained by direct sum decomposition of the state space taking s as a projection subspace ($X = X_1 + X_2$) with X_1 being the *projection of X on s along Pb* and X_2 the *projection of X on Pb along s*. The time derivative of $V(\underline{x})$ contains two clearly identified terms: one dealing with the sliding regime creation process ($2s\,ds/dt/\underline{b}^TP\underline{b}$) and the other intimately related to the ideal sliding dynamics stability:

$$\underline{x}^T[(A-\underline{b}\underline{b}^TPA/\underline{b}^TP\underline{b})^TP+P(A-\underline{b}\underline{b}^TPA/\underline{b}^TP\underline{b})]\underline{x}$$

Design Example

Consider the dynamical system described by: $dx_1/dt = x_2$; $dx_2/dt = x_3$; $dx_3/dt = -0.0226x_1-2.262x_2-0.21x_3 + u$.

An infinite time performance index without control penalty defined by $Q = \mathrm{diag}\,[10,1,1]$ and a surface equation $s = \underline{b}^T P \underline{x} = 20.0x_1 + 8.0x_2 + x_3 = 0$ yields a VSC with two feedback loops and magnitude bounds for the feedback gains given by $m_1 = 155.776$, $m_2 = 44.58$. The ideal sliding dynamics is assymptotically stable. Fig.1 shows a sample of the state trajcetory in the open loop case and Fig. 2 shows the corresponding VSC feedback response.

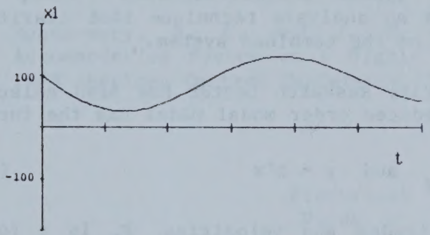


Fig.1

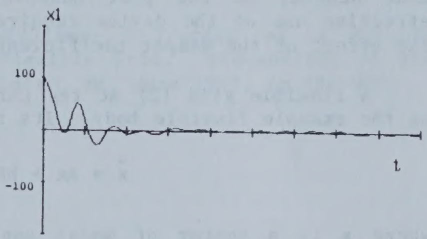


Fig.2

CONCLUSIONS

The relevance of Lyapunov's Second Method for the design of VSS for linear stationary plants has been established. A simple Lyapunov function decomposition ,based on geometric arguments induced by considering the switching surface as a projection subspace, leads to a separation of the sliding mode formation process and the ideal sliding dynamics stabilization. The approach was shown to have strong connections with optimal feedback design via Krasovskii's sinthesis procedure . A simple third order design example was also furnished.

A number of problems within the VSS area can be further analyzed from this viewpoint: Multivariable systems, time-varying systems, non-linear systems and the non-linear sliding manifold case [3].

REFERENCES

[1] Utkin,V.I.,Sliding Modes in Variable Structure Systems MIR. Publishers. Moscow 1978

[2] Csaki,F.,Modern Control Theories.Akademiai Kiado. Budapest. 1972.

[3] Sira-Ramirez,H., "Non-linear Sliding Manifolds for Linear and Bilinear Systems" 24th IEEE Conference on Decision and Control.Fort Lauderdale,Florida. December 1985 (to appear).

ANALYZING INTERCONNECTED SYSTEMS USING THE DUAL GENERALIZED HESSENBERG REPRESENTATION (GHR)

DOUGLAS K. LINDNER
VPI&SU
Blacksburg, VA

RAYMOND C. MONTGOMERY
NASA Langley Research Center
Hampton, VA

The Dual GHR [1] is a linear system representation that is useful in analyzing interconnected systems. This paper illustrates its use in analyzing a vibration damper attached to a flexible body. The principal of operation of the damper (see the figure) is that motions of its case of mass m_c cause a force between the case and the proof mass, m_p, because of a viscous fluid. A valve in the fluid circuit allows the coefficient of damping, c, to be varied. One operational use for this is to close off the valve (thus, locking the proof mass) during maneuvers or articulation of structural members. Afterwards, the valve is set to an intermediate value that depends on the post maneuver system characteristics. Thus, the effective use of the device requires an analysis technique that clarifies the effect of the damper coefficient on the combined system.

A flexible grid [2] at the Langley Research Center has been selected as the example flexible body. Its reduced order modal model has the form

$$\ddot{x} = Ax + bF_g \quad \text{and} \quad y = b'x \tag{1}$$

where x is a vector of modal amplitudes and velocities, F_g is a force input, and y is the grid displacement at the damper attachment point. Using the symbols defined in the figure, the equations of motion for the damper are:

$$m_p\ddot{x}_p = -k(x_p - x_c) - c(v_p - v_c) \tag{2}$$

$$m_c\ddot{x}_c = -k(x_c - x_p) - c(v_c - v_p) + F \tag{3}$$

When attached to the grid $F = -F_g$ and $x_c = y$. With (3), these constraints produce a combined system dynamics of:

$$(I+bb'm_c)\ddot{x} = (A-kbb')\dot{x} - cbb'x + cb\dot{x}_p + kbx_p \tag{4}$$

The system represented by (2) and (4) can be used to study the combined grid/damper poles as a function of the damping parameter c. The Dual GHR form, wherein bb' becomes a matrix with only one non-zero element, facilitates this analysis. Specifically, using this form, the system characteristic polynomial can easily be written [1] as:

$$p_d(s) = (s^2 + .56cs + .56)(s^2 + 8.59) - 3.38(cs + 1)$$

$$p_g(s) = (s^2 + 41.17)(s^2 + 22.15) - (24.53)^2$$

$$p(s) = p_d \cdot p_g - 1.74(7.73)(s^2 + .17cs + .17)(s^2 + 22.15)$$

The polynomials p_d and p_g are respectively associated with the Dual GHR representations of the damper and the grid. The numerical values correspond to the 1/8 in. Langley grid and a damper with a 2 lbm case, a 6 lbm proof mass, and a spring of 1 lbf/ft. Examination of p(s) shows that it can be put in root locus form where c is the variable gain. This is a direct result of the Dual GHR form.

In conclusion, this paper has presented an example of the use of the Dual GHR. The method involves a canonical form that is extremely useful in making visible the characteristic polynomial of relatively complex interconnected systems.

References

1. Lindner, D. K.: The Dual GHR. Proceedings of the Allerton Conference on Communications, Control, and Computing, Monticello, IL, Oct 1984, pp 745-752.

2. Montgomery, R. C. and J. P. Williams: Testing of A Failure Accommodation System on A Highly Flexible Grid. Proceedings of the 1985 American Control Conference, Boston, MA, June 1985, pp 984-989.

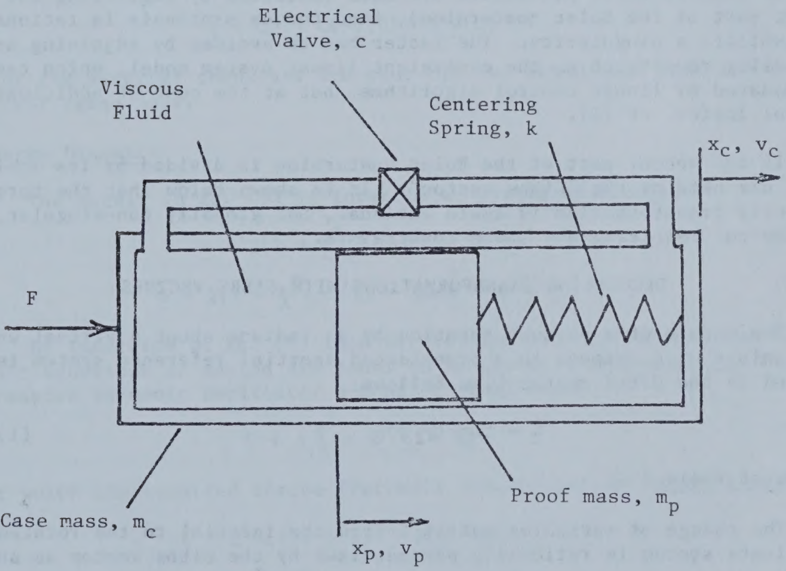


Figure - The variable coefficient viscous inertial damper.

GIBBS VECTOR KINEMATICS AND DECOUPLED ATTITUDE MANEUVERS

THOMAS A. W. DWYER, III*
University of Illinois at Urbana-Champaign, Illinois

ABSTRACT

By use of Gibbs vectors as kinematic variables, multi-axial spacecraft equations of rotational motion are shown to be globally and rationally feedback-equivalent to a set of unconstrained and decoupled harmonic oscillators, driven by generalized acceleration commands.

CHOICES OF KINEMATIC VARIABLES

When Euler angle sequences are used as kinematic variables, it is possible to invert the equations of attitude dynamics so that torque profiles are synthesized from commanded Euler angle accelerations, with state-dependent automatic gain scheduling. The transformations are, however, transcendental and possibly singular.

If Euler quaternions are used instead for nonsingular quadratic gain scheduling, an equality constraint is required in the attitude variables, which cannot be handled by linear control algorithms.

If reduced Euler parameters are used (obtained by neglecting the scalar part of the Euler quaternion), then torque synthesis is rational, but exhibits a singularity. The latter can be avoided by adjoining an inequality constraint to the equivalent linear system model, which can be accomodated by linear control algorithms, but at the cost of additional control logic: cf [1].

If the vector part of the Euler quaternion is divided by its scalar part, one obtains the "Gibbs vector". It is shown below that the torque synthesis transformation is again rational, but globally non-singular, thereby not requiring any state constraints.

DECOUPLING TRANSFORMATIONS WITH GIBBS VECTORS

The result of a virtual rotation by ϕ radians about a virtual unit skew axis $\underline{e}$ with respect to a preselected inertial reference system is encoded in the Gibbs vector $\underline{\xi}$ as follows:

$$\underline{\xi} = \tan(\tfrac{1}{2}\phi)\underline{e} \qquad (1)$$

Change of Basis

The change of variables matrix C from the inertial to the rotated coordinate system is rationally parametrized by the Gibbs vector as shown below,

$$C(\underline{\xi}) = 2(1 + \underline{\xi}^T\underline{\xi})^{-1}[I + \underline{\xi}\underline{\xi}^T - \underline{\xi}x] - I \qquad (2)$$

where I is the 3 x 3 identity matrix and $[\xi x]$ the matrix defining the

---

* Supported by NSF Grant ECS-8516445 and by NASA Grant NAG-1-436.

vector product operation in the rotated coordinate system, as follows by transforming the quaternion parametrization in [1]. One has:

$$C(\underline{\xi})^{-1} = C(\underline{\xi})^{T} = C(-\underline{\xi}) \tag{3}$$

$$C(\underline{\xi}_1)\ C(\underline{\xi}_2) = C(\underline{\xi}_1 * \underline{\xi}_2),\ \underline{\xi}_1 * \underline{\xi}_2 = (1 - \underline{\xi}_1^{T}\ \underline{\xi}_2)^{-1}\ (\underline{\xi}_1 + \underline{\xi}_2 + \underline{\xi}_1 \mathrm{x}\ \underline{\xi}_2) \tag{4}$$

Kinematics

If $\underline{\omega}$ denotes the spacecraft angular velocity in spacecraft coordinates then the kinematic equation, transformed from the quaternion kinematics given in [1] and elsewhere, becomes

$$\dot{\underline{\xi}} = \tfrac{1}{2}\,[I + \underline{\xi}\underline{\xi}^{T} + \underline{\xi}\mathrm{x}]\ \underline{\omega} \tag{5}$$

Dynamics

If $\underline{h}$ denotes the total spacecraft angular momentum in spacecraft coordinates, I° the total central inertia matrix, I^{a} the matrix of axial wheel inertias, τ° the external torques, $\underline{\mu}$ the reaction wheel torques then

$$J\dot{\underline{\omega}} = \underline{h}\ \mathrm{x}\ \underline{\omega} + \underline{\tau} \tag{6}$$

with $J = I^{\circ}$, $\underline{h} = I^{\circ}\ \underline{\omega}$ and $\underline{\tau} = \underline{\tau}^{\circ}$ while the wheels are locked, then with $J = I^{\circ} - I^{a}$, $\underline{\tau} = -\underline{\mu}$ and

$$\underline{h} = C(\underline{\xi})\ C(-\underline{\xi}(0))\ I^{\circ}\underline{\omega}(0) \tag{7}$$

after the external thrusters are shut off and orthogonal reaction wheel control takes over.

Inverse Dynamics

The matrix in Eq. (5) is invertible, giving eventually

$$\underline{\omega} = 2(1 + \underline{\xi}^{T}\underline{\xi})^{-1}\ [I - \underline{\xi}\mathrm{x}]\dot{\underline{\xi}} \tag{8}$$

$$\dot{\underline{\omega}} = 2(1 + \underline{\xi}^{T}\underline{\xi})^{-1}\ [I - \underline{\xi}\mathrm{x}]\ddot{\underline{\xi}} - \underline{\omega}\ \underline{\omega}^{T}\underline{\xi} \tag{9}$$

By insertion of Eq. (9) into Eq. (6) and simplification, the spacecraft equations of motion are found to be globally equivalent to the decoupled harmonic oscillator autopilot model below,

$$\ddot{\underline{\xi}} + a_1\dot{\underline{\xi}} + a_o\,\underline{\xi} = b\underline{u} \tag{10}$$

for which the required torque synthesis transformation is as follows:

$$\begin{aligned} \underline{\tau} &= 2\ b(1 + \underline{\xi}^{T}\underline{\xi})^{-1}\ J[I - \underline{\xi}\mathrm{x}]\ \underline{u} \\ &\quad - J\{2a_o(1 + \underline{\xi}^{T}\underline{\xi})^{-1}\underline{\xi} + a_1\underline{\omega} + \underline{\omega}\ \underline{\omega}^{T}\ \underline{\xi}\} + \underline{\omega}\ \mathrm{x}\ \underline{h} \end{aligned} \tag{11}$$

REFERENCE

[1] T.A.W. Dwyer, III and A.L. Batten, "Exact Spacecraft Detumbling and Reorientation Maneuvers with Gimbaled Thrusters and Reaction Wheels", J. of the Astronautical Sciences v. 33, No. 2 (April-June 1985), pp. 217-232.

ANALYSIS OF TRANSIENT OSCILLATIONS IN A NONLINEAR FEEDBACK SYSTEM

R. K. Miller*
Department of Mathematics
Iowa State University
Ames, Iowa 50011

A. N. Michel**
Department of Electrical Engineering
University of Notre Dame
Notre Dame, Indiana 46556

ABSTRACT

In several recent papers [1]-[4] we established rigorous results which indicate when the describing function method is valid (i.e., under which conditions it predicts correctly the existence and the stability of a limit cycle). These results yield also estimates of bounds for limit cycles during transient time periods (i.e., during limit cycle build up or limit cycle build down). In the present paper we address the accuracy of our earlier results. In doing so, we analyzed several typical feedback systems by the method of [1] and we compared these results with those obtained via numerous simulations. Due to space limitations, the results of only one representative feedback system are included.

INTRODUCTION

In several recent papers [1]-[4], rigorous results were obtained which make it possible to predict correctly the existence of limit cycles in feedback systems as well as their stability properties. These results yield also estimates of bounds during limit cycle build up or build down. In addition, these results show when and why the usual quasistatic method of analysis (i.e., the Loeb criterion [5]) may incorrectly predict the stability properties of a limit cycle.

Although the results in [1]-[4] provide a rigorous foundation for describing function methods, they still constitute only approximations to true solutions of feedback systems. For this reason we investigated several typical feedback systems, using extensive simulations, to find out exactly how accurate our results in [1]-[4] are. In the present paper, which focuses primarily on the development in [1], we present the results of our findings for one such feedback system (Figure 1). During this study we discovered that some of our results for limit cycle build up and limit cycle build down, which agree with actual simulations, are in variance with earlier results dealing with this topic ([5, section 4.3]).

The remainder of this paper is organized as follows. In Section II we present an analysis of the system of Figure 1, based on [1]. In Section III we compare the results for a specific example of Figure 1 obtained by the methods of Section II with those obtained by means of extensive numerical simulations. We conclude the paper in Section IV with some concluding comments.

THE SYSTEM AND ITS ANALYSIS

We consider a typical nonlinear feedback system shown in Figure 1 consisting of a saturation nonlinearity, a second order plant and a linear com-

---

[*] The work of R. K. Miller was supported in part by the National Science Foundation under Grant ECS 85-01884.

[**] The work of A. N. Michel was supported in part by the National Science Foundation under Grant ECS 84-19918.

pensator. If the method of describing functions is applied to this system (see, e.g. [5]), then one must find $\omega>0$ and $A>\delta$ such that

$$\frac{s^2+2\xi\omega_n s+\omega_n^2}{\omega_n^2} + \frac{(1+\tau s)}{s} kN(A) = 0 \ , \ s = j\omega \qquad (1)$$

where $N(A)$ denotes the sinusoidal-input describing function for the given saturation function, i.e.,

$$N(A) = \begin{cases} m \text{ if } A \leqslant \delta \\ \frac{2m}{\pi}\{\sin^{-1}(\delta/A) + (\delta/A)[(1-(\delta/A)^2]^{1/2}\} \text{ if } A > \delta \ . \end{cases} \qquad (2)$$

In order for such a solution to exist, it is necessary that

$$2\xi\omega_n[(mk)^{-1} + \tau] < 1.$$

We shall assume that this is the case. In fact, we shall assume that (1) has been solved, either graphically or numerically, and that a solution $A_0>\delta$ and $\omega_0>0$ has been obtained.

Now define

$$q(s) = s^3 + 2\xi\omega_n s^2 + \omega_n^2 s \ , \ p(s) = k\omega_n^2(1 + \tau s)$$

and

$$d(s) = q(s) + p(s)N(A_0) \ .$$

Since $d(\pm j\omega_0) = 0$, then $d(s)$ can be factored as

$$d(s) = (s^2 + \omega_0^2)(s + 2\xi\omega_n) \ .$$

Next, define two complex numbers β and Δ by

$$\beta = \beta_1 - j\beta_2 = \frac{2}{\omega_0 d'(j\omega_0)}$$

and

$$\Delta = \Delta_1 + j\Delta_2 = (\beta_1 - j\beta_2)p(j\omega_0)$$

and let $\varepsilon = |\beta| = (\beta_1^2 + \beta_2^2)^{1/2}$.

Using several novel transformations to put a given system into an appropriate form, and applying the stable manifold theorem, it can be shown by the results of [1] that when $\varepsilon>0$ is sufficiently small, then the output $y(t)$ of the system of Fig. 1 will have the form

$$\begin{aligned} y(t) &= [A_0 + Er(t)] \sin(\omega_0 t + F + \theta(t)) \\ &\quad + \sqrt{\varepsilon}\, B\, x_1(t) + \mathcal{O}(\varepsilon) \end{aligned} \qquad (3)$$

where B is a fixed constant, E and F are real constants such that $E\exp(jF) = p(j\omega_0)$, $\mathcal{O}(\varepsilon)$ denotes terms which are of order ε and $r(t)$, $\theta(t)$ and $x_1(t)$ are solutions of a system of differential equations of the form

$$\dot{r} = \frac{\Delta_1\omega_0}{2E}(Er + A_0)[N(A_0) - N(Er + A_0)] + \text{"small"}$$

$$\dot{\theta} = \frac{\Delta_2\omega_0}{2}[N(A_0) - N(Er + A_0)] + \text{"small"} \tag{4}$$

$$\dot{x}_1 = -2\xi\omega_n\omega_0 x_1 + \mathcal{O}(\sqrt{\varepsilon}).$$

Here "small" denotes perturbation terms whose absolute values are small when ε, $|x_1|$ and $|r|$ are small.

If the small perturbation terms in (4) are dropped and the resulting equations are linearized, we obtain the set of differential equations

$$\dot{r} = -\frac{\Delta_1\omega_0 A_0}{2}N'(A_0)r \tag{5a}$$

$$\dot{\theta} = -\frac{\Delta_2\omega_0 E}{2}N'(A_0)r \tag{5b}$$

$$\dot{x}_1 = -2\xi\omega_n\omega_0 x_1 \;. \tag{5c}$$

When ε is small, system (4) will have an oscillation near the predicted limit cycle

$$y_0(t) = A_0\sin(\omega_0 t + F).$$

This limit cycle will be stable if $\xi>0$ so that (5c) is stable and $\Delta_1>0$ so that (5a) is stable. The property that $\Delta_1>0$ is equivalent to the usual graphical crossing criterion (see, e.g., [5]) which is also called the Loeb criterion (or the quasistatic stability analysis). It should be noted however, that equation (5) yields even more detailed information than the usual describing function analysis (Loeb criterion). To see this, note that the constant $c \overset{\Delta}{=} 2\xi\omega_n\omega_0$ is typically much larger than the constants $a \overset{\Delta}{=} -\Delta_1\omega_0 A_0 N'(A_0)$ and $b \overset{\Delta}{=} -\Delta_2\omega_0 E N'(A_0)$. Thus, equation (3) predicts that the output $y(t)$ will approach a small neighborhood of the steady state oscillation since $|x_1(t)|$ will decay rapidly. Since $r(t) \rightarrow 0$ and $\theta(t) \rightarrow \theta_\infty$ more slowly, then once $y(t)$ reaches this small neighborhood, it will require a long time duration to finish settling out. The rates of decay of $r(t)$ and $x_1(t)$ can be estimated from (5). Moreover, since approximately, $\dot{\theta} = -br = -br_0\exp(-at)$, then

$$\theta(t) \cong \theta(0) - \frac{br_0}{a} + \frac{br_0}{a}e^{-at}\;. \tag{6}$$

Hence, the phase shift in $y(t)$ will speed up or slow down during the final stage of the settling according to whether br_0/a is positive or negative.

A SPECIFIC EXAMPLE

To determine how good the predictions obtained by means of the results of Section II really are, the system of Fig. 1 was solved numerically for various values of the system parameters. We shall present here some results for the case where $\tau = 1/20$, $m = 1$, $k = 30$, $\omega_n = 1$, $\xi = 4$ and $\delta = 1/10$.

These values were chosen for illustration purposes and do not reflect values for any particular physical system. For these values a single stable limit cycle is predicted with

$$A_0 = 0.280279228 \ , \ N(A_0) = 4/9$$

$$\omega_0 = \sqrt{5/3} = 1.290994449$$

and the predicted limit cycle period is

$$T_0 = 2\pi\sqrt{3/5} = 4.866934411.$$

Moreover,

$$\beta_1 = -0.0117958884 \quad \text{and} \quad \beta_2 = .0730964467$$

and these yield $\varepsilon \cong 0.0740$. Experience indicates that this ε is small enough for the present theory to yield correct predictions.

For the present example, equation (5) assumes the form

$$\dot{r} \cong -0.0582r \tag{7a}$$

$$\dot{\theta} \cong -65.103r \tag{7b}$$

$$\dot{x}_1 \cong -10.3280x_1. \tag{7c}$$

Numerical simulations indicate a globally orbitally asymptotically stable limit cycle

$$y_1(t) \cong A_1\sin(\omega_1 t)$$

where

$$A_1 \cong 0.2962, \ \omega_1 \cong 1.244$$

and hence, an actual period of

$$T_1 \cong 5.052.$$

Hence, the error in the predicted amplitude is less than 6% while the error in the predicted period is less than 4%.

From equation (7c) we predict that the "fast transient" $x_1(t)$ will be essentially negligible when $t > 0.5$. In our numerical studies the fast transient seems to disappear very rapidly (see Figures 2, 3 and 4), although there is no way to directly compute $x_1(t)$ and verify this precisely.

According to equation (6), the phase $\theta(t)$ should be increasing when $y(t)$ is initially smaller than A_1 and it should be decreasing when $y(t)$ is initially larger than A_1. Figure 2 shows a typical example of limit cycle build up. Using a ruler, one can readily verify that $\theta(t)$ is indeed increasing. On the other hand, in examples where $y(t)$ starts large, $\theta(t)$ can be seen to decrease. Also note that from (7a) and (7b) one would predict that $\theta(t)$ tends to a limit θ_∞ as $t \to \infty$ and that

$$\theta(t) - \theta_\infty \cong (\text{constant}) \times \exp(-0.0582t).$$

A detailed analysis of the numerical results used to generate Figure 2 indicates that in fact

$$\theta(t) - \theta_\infty \cong (\text{constant}) \times \exp(-0.0720t).$$

Hence, the error in the predicted rate constant 0.0582 seems to be about 20%.

Equation (7a) indicates a decay rate for r of 0.0582. The actual decay rate of r (when r is nearly zero) seems to be about 0.0720, a 20% error. However, this decay rate will vary significantly if r is very far from zero. According to equation (3), the amplitude $A(t)$ of $y(t)$ satisfies $A(t) \cong Er(t) + A_1$. When $A(t)$ is near zero, then the saturation function is in its linear range and equation (7a) constitutes a very poor approximation. For $A(t) > 0.1 = \delta$, the decay rate increases with A. When $A(t) \cong 0.15$, the decay rate is about 0.062, a 7% error; when $A(t) \cong 0.25$, the decay rate is about 7% a 20% error; when $A(t) \cong 0.35$, the decay rate is about 0.075, a 29% error; and for large values of $A(t)$, the rate goes up to about 0.081, a 40% error. This behavior can be observed in Figures 3 and 4.

Figure 3 shows a limit cycle build up. The curve $y(t)$ is graphed as well as the bounding curves

$$x = \pm[0.296 - 0.472 \exp(-0.072t)].$$

The exponential rate 0.072 does not fit the beginning of the limit cycle build up at all. However, it fits approximately at midrange and very well at the end when the amplitude of $y(t)$ is near $A_1 \cong 0.296$.

Figure 4 depicts $y(t)$ building down to the limit cycle. The bounding curves are

$$x = \pm[0.296 + 0.620 \exp(-0.078t)].$$

The exponential decay rate 0.078 is seen to fit the beginning portion of the curve rather well but is too large over the final portion. Notice also that the phase shift $\theta(t)$ is decreasing in this plot as we have predicted.

A plausible explanation for the inaccuracies in the rate of decay of $r(t)$ for small values of the amplitude is the following. When the small perturbation term in the first equation of (4) is dropped, the result is

$$\dot{r} = \frac{\Delta\ \Delta_1\omega_0}{2E}(Er + A_0)[N(A_0) - N(Er + A_0)]. \tag{8}$$

When this equation is linearized to obtain (5a), the linearized equation is a reasonably accurate approximation of (8) for $Er + A_0$ larger than A_0 but not for $Er + A_0$ smaller than A_0.

If all parameters except k are held constant, then the predicted amplitude $A_0(k)$ of the limit cycle will vary with k. We have chosen $k = 30$ for illustration purposes. As k decreases, it was found that $A_0(k)$ will decrease until $A_0(k)$ reaches $\delta = 0.1$ at $k = 40/3 = 13\ 1/3$. For $k < 40/3$, no limit cycle exists. As k decreases, equation (5a) becomes a poorer predictor of the rate of limit cycle build up. For example, for $k = 14$ we obtain $A_0(k) = 0.1133425$. Obviously, equation (5a) cannot possibly be valid until $r(t)$ is in the range $|r(t)| < 0.013$; however, the unlinearized equation (8) will provide somewhat more accurate predictions.

CONCLUSIONS

The above results for the feedback system of Figure 1, as well as results for several other typical feedback systems indicate that the methodology advanced in [1] yields correct and reasonably accurate estimates.

ACKNOWLEDGEMENT

The authors wish to thank Mr. J. A. VanKley for generating Figures 2, 3 and 4 as well as many other figures for this study which are not displayed here.

REFERENCES

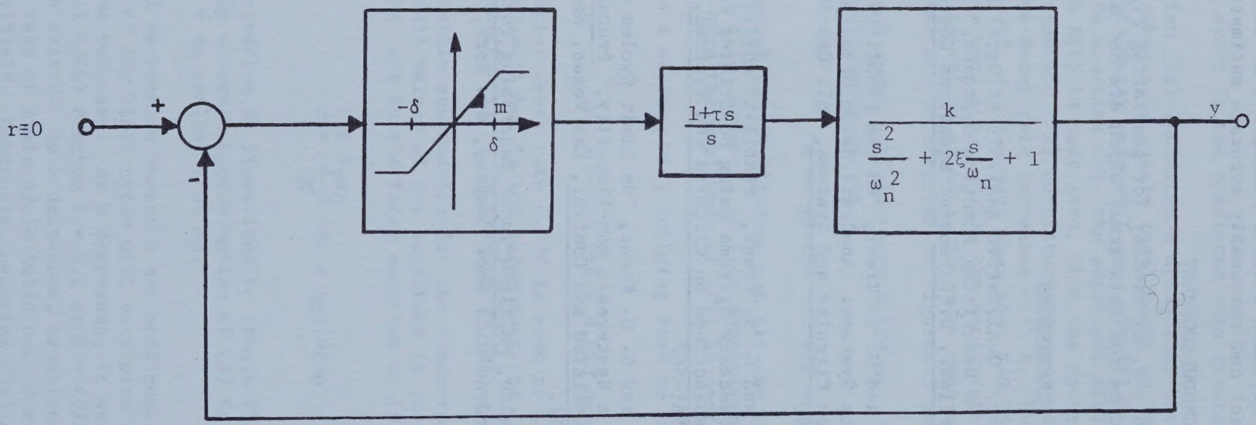
[1] G. S. Krenz and R. K. Miller, "Qualitative Analysis of Oscillations in Nonlinear Control Systems: A Describing Function Approach," Proceedings of the 1984 Midwest Ordinary Differential Equations Conference (to appear)

[2] R. K. Miller, A. N. Michel and G. S. Krenz, "On the Stability of Limit Cycles in Nonlinear Feedback Systems: Analysis Using Describing Functions," IEEE Transactions on Circuits and Systems, Vol. CAS-30, No. 9, September 1983, pp. 684-697.

[3] R. K. Miller, A. N. Michel and G. S. Krenz, "Stability Analysis of Limit Cycles in Nonlinear Feedback Systems Using Describing Functions: Improved Results," IEEE Transactions on Circuits and Systems, Vol. CAS-31, No. 6, June 1984, pp. 561-567.

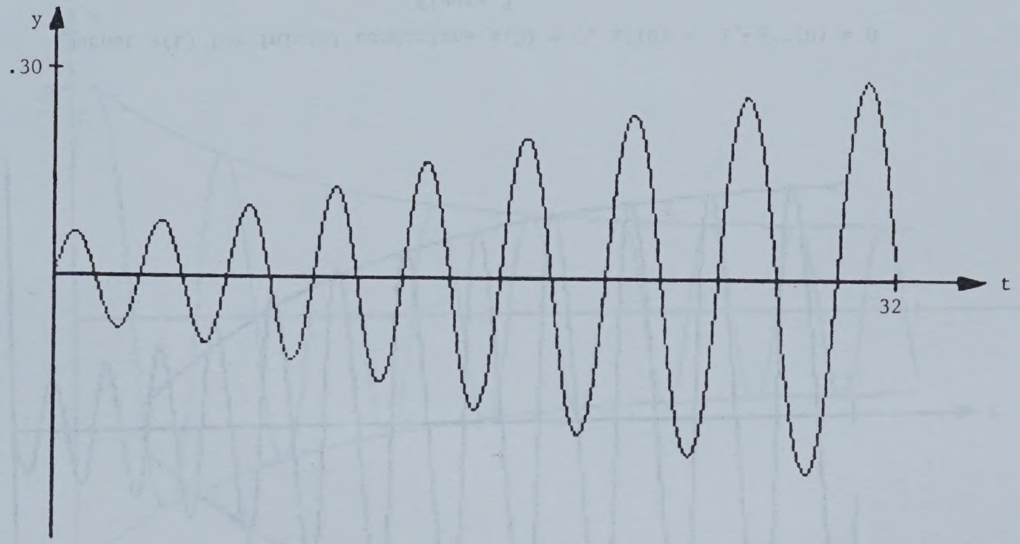
[4] R. K. Miller, A. N. Michel and G. S. Krenz, "On Limit Cycles of Feedback Systems which Contain a Hysteresis Non-linearity," Proceedings of the 23 IEEE Conference on Decision and Control, Las Vegas, Nevada, December 1984, pp. 1306-1311.

[5] A. Gelb and W. E. Vander Velde, Multiple-Input Describing Functions and Nonlinear System Design, McGraw-Hill Book Company, New York, 1968.


r≡0
+
-
-δ
δ
m
1+τs
s
k
s^2 / ω_n^2 + 2ξ s/ω_n + 1
y

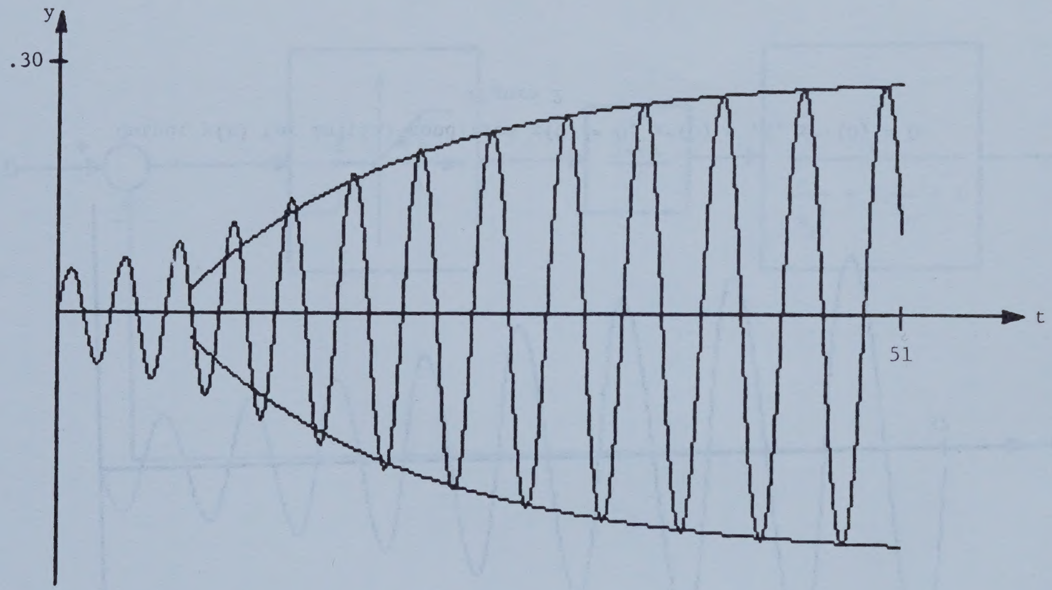
Control System

Figure 1



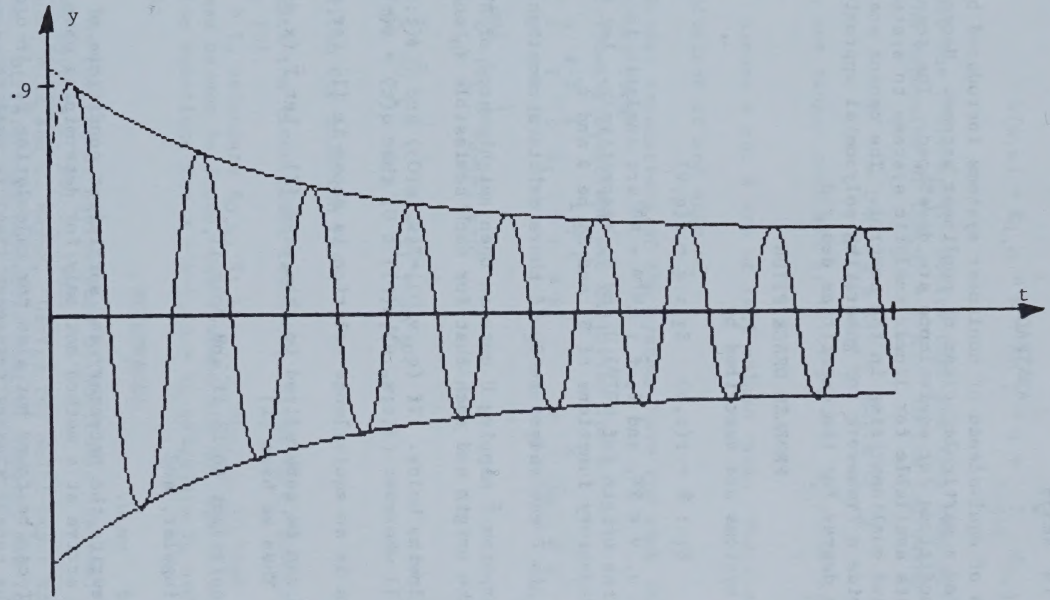
Output $y(t)$ for initial condition $x(0) = 0$, $x'(0) = .1$, $x''(0) = 0$

Figure 2



Output y(t) for initial conditions x(0) = 0, x′(0) = .1, x′′(0) = 0

Figure 3



Output $y(t)$ for initial conditions $x(0) = 0$, $x'(0) = .6$, $x''(0) = 0$

Figure 4

A THEOREM ON THE FEEDBACK EQUIVALENCE OF NONLINEAR SYSTEMS USING TENSOR ANALYSIS

JOSEPH A. O'SULLIVAN AND MICHAEL K. SAIN
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

ABSTRACT

A definition of equivalence of nonlinear systems introduced by Su [1] is examined for a particular class of nonlinear sytems. Necessary and sufficient conditions for equivalence are developed. The results here extend some results available for linear analytic systems to systems with multiple inputs and nonlinearities in the control. The tensor analytic methods used provide a framework for generating polynomial approximations of an appropriate degree for the quantities desired.

PROBLEM DESCRIPTION

Suppose two systems are described by

$$S_1 : \dot{x} = f(x,u) \qquad S_2 : \dot{q} = \tilde{f}(q,v)$$

where x, q ε R^n , u, v ε R^m, and f, $\tilde{f}$: $R^{n+m} \rightarrow R^n$ are analytic in open neighborhoods of the origin (of R^{n+m}) U and V, respectively. Let the respective state trajectory functions of S_1 and S_2 be ϕ and ψ.

Definition [2]:S_1 is $\tilde{T}$-equivalent to S_2 if there exists a nonsingular analytic diffeomorphism $\tilde{T}$ mapping U onto an open neighborhood of R^{n+m} which preserves the origin and such that for each admissible x_o and control u the following holds. If $(q_o,v(0))=\tilde{T}(x_o,u(0))$ and $\tilde{T}(\phi(t;x_o,u), u(t)) = (q(t),v(t))$ whenever $(\phi(t;x_o,u),u(t))$ ε U, then $q(t) = \psi(t;q_o,v)$.

The fact that this is an equivalence relation is shown in [1] for single-input systems and can be generalized to this case [2]. Let $\tilde{T}$ (x,u) = (T(x,u),k(x,u)). Then we have [1]

Lemma: S_1 is $\tilde{T}$-equivalent to S_2 if and only if
a. $\partial T/\partial u = 0$,
b. $\partial k/\partial u$ is nonsingular, and
c. $\tilde{f}\circ\tilde{T} = (\partial T/\partial x)f$ on U.

We wish to rewrite the necessary and sufficient conditions of the lemma in order to arrive at a method not only for determining when such a transformation $\tilde{T}$ can be found but also for calculating $\tilde{T}$. In order to carry this out, the tensor algebra framework [3-4] is used to expand the system and transformation functions in series of homogeneous powers of the variables. These series are substituted into the conditions of the lemma. The resulting equations give insight into the nature of this equivalence and provide a computationally attractive setting to find the transformation tensors.

MAIN RESULT

Because f, $\tilde{f}$, and $\tilde{T}$ are analytic they can be expanded as

$$f(x,u) = Ax + Bu + \sum_{i+j=1}^{\infty} A_{i,j} \overset{i+j}{\odot} x^i \otimes u^j$$

$$T(x) = T_1 \overset{1}{\odot} x + \sum_{i=2}^{\infty} T_i \overset{i}{\odot} x^i$$

$$k(x,u) = K_{1,0} \overset{1}{\odot} x + K_{0,1} \overset{1}{\odot} u + \sum_{i+j=2}^{\infty} K_{i,j} \overset{i+j}{\odot} x^i \otimes u^j$$

The tensors for $\tilde{f}$ are $\tilde{A}_{i,j}$. Notice that the fact that $\partial T/\partial u = 0$ has been used in the expansion of $\tilde{T}(x,u) = T(x)$. In the following

$$K^{j}_{i,\ell} = \Sigma K_{i_1,\ell_1} \otimes K_{i_2,\ell_2} \otimes \ldots \otimes K_{i_j,\ell_j}$$

where the sum ranges over all $i_1 + \ldots + i_j = i$, $\ell_1 + \ldots + \ell_j = \ell$.

Theorem: Assume B and $\tilde{B}$ are of full column rank. The systems S_1 and S_2 are $\tilde{T}$-equivalent if and only if

a. The linear truncations of the systems have the same Brunovsky canonical form

b. The following equations are satisfied for $k > 1$:

$$\alpha \neq 0 : \pi_s\Big(\sum_{\ell=1}^{\alpha} \sum_{j=1}^{k-\ell} \sum_{i=1}^{\ell} \tilde{A}_{i,j} \overset{i+j}{\odot} T^{i}_{\ell} \otimes K^{j}_{\alpha-\ell,k-\alpha} + \sum_{j=1}^{k} \tilde{A}_{0,j} \overset{j}{\odot} K^{j}_{\alpha,k-\alpha}\Big) =$$

$$\pi_s\Big(\sum_{i=1}^{\alpha+1} iT_i \overset{1}{\odot} A_{\alpha-i+1,k-\alpha}\Big)$$

$$\alpha = 0 : \pi_s\Big(\sum_{j=1}^{k} \tilde{A}_{0,j} \overset{j}{\odot} K^{j}_{0,k}\Big) = T_1 \overset{1}{\odot} A_{0,k}$$

The algorithm for solution is:

1. Calculate the first order terms (T_1, $K_{1,0}$, $K_{0,1}$) from the linear system theory [5].
2. For $k > 1$, calculate $K_{0,k}$, $K_{1,k-1}, \ldots, K_{k-2,2}$, in that order. If any of these cannot be found, the systems are not equivalent.
3. From the equations for $\alpha = k-1$, $\alpha = k$, calculate T_k and $K_{k,0}$ and $K_{k-1,1}$.

REFERENCES

1. R.Su, "On the Linear Equivalents of Nonlinear Systems," Systems and Control Letters 2, No. 1, 1982.
2. L.R.Hunt, R.Su and G. Meyer, "Design for Multi-Input Nonlinear Systems," Differential Geometric Control Theory, New York: Birkhauser, pp. 268-298, 1982.
3. J.A. O'Sullivan and M.K. Sain, "Nonlinear Optimal Control with Tensors: Some Computational Issues," Proc. ACC, Boston, pp. 1600-1605, 1985.
4. M.R. Buric, "Application of Multilinear Algebra in Optimal Regulation of Nonlinear Polynomial Systems," Proc. 16th Allerton Conference on Communication, Control, and Computing, Urbana, IL, Univ. of IL, pp. 655-663, 1978.
5. R.W. Brockett, "Feedback Invariants for Nonlinear Systems," Proc. IFAC Congress, Helsinki, pp. 1115-1120, 1978.

LOGICAL FAULT ANALYSIS AND DESIGN FOR TESTABILITY OF PROGRAMMABLE LOGIC ARRAYS

PRADIP BOSE
IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

ABSTRACT

Improved procedures for fault analysis and testing of PLAs are presented. Single and multiple contact faults are considered. Improved upper bounds on test set sizes are achieved as a result of our methodology. A procedure to design partially irredundant PLAs with a view to reducing test generation complexity is sketched.

I. INTRODUCTION

Programmable logic arrays (PLAs) have been receiving increasing recognition as versatile, cost-effective modules in large-scale logic design. Consequently, research papers in the areas of testing (e.g., [1-6]) and design for testability (e.g., [7,8]) of PLAs have mushroomed, as have those in the areas of PLA design and minimization.

In this paper, we first analyze physical faults of a PLA in terms of a simple, logical fault model as proposed in [2-5, 8]. Failures are viewed entirely as changes or deformations of the product terms of the PLA. Both single and multiple faults are considered. A very simple, heuristic method for generating a comprehensive test set for a PLA is proposed. This procedure is much faster than previously reported methods. At the same time, the test set sizes are very small, and are often minimal. Based on these results, we propose a method for designing easily testable PLAs. This latter method is not based on using extra control inputs as in [8] or modifying the basic PLA circuitry as in [7]. It is simply a procedure for optimally choosing and placing the product terms in order to implement a set of specified Boolean functions; such a design ensures a drastic reduction in the test generation complexity.

II. THE CROSSPOINT (CONTACT) FAULT MODEL

For the purposes of this paper, we restrict ourselves to the crosspoint fault model traditionally used for PLAs [1-3], in which a single fault causes a programmed 0(1) in the input or output array to change to a 1(0). It has been shown [2-5,8] that this physical fault model is equivalent to a logical fault model in which the product terms are subject to growth (G), shrinkage (S), disappearance (D), and appearance (A) faults. An additional type of fault, (V) in which a product term vanishes totally from all the output functions it belongs to, should be considered for completeness as pointed out in [5,8]. However, for testing purposes, since tests for S faults always cover V faults, the latter need not be considered explicitly.

III. PRELIMINARY NOTATIONS AND DEFINITIONS

A PLA with n variables, m product terms and p output functions is referred to here as a (n,m,p)-PLA. This general PLA organization has m(2n+p) crosspoints or contacts.

We use an (n+p)-tuple notation for the rows (cubes) of a (n,m,p)-PLA. This notation is of the form $(y_1y_2...y_n, z_1z_2...z_p)$, with $y_i \in [0,1,x]$ and $z_j \in [0,1]$. The n-tuple $(y_1y_2...y_n)$ is the product term (or, "implicant," or "input part," or simply, "term"), and the p-tuple $(z_1z_2...z_p)$ is the corresponding output connection vector (OCV), or "output part." We will occasionally denote the input part of a cube c as $c_{i.p}$ and the corresponding output part as $c_{o.p}$.

A growth (G) term is an n-tuple $G^{(n)} = (g_1g_2...g_n)$, $g_i \in [0,1,x]$, denoting the set of extra minterms contributed by a G-fault. An appearance (A) term is defined similarly.

A shrinkage (S) term is an n-tuple $S^{(n)}$ denoting the set of minterms which disappear on a S-fault.

A disappearance (D) term is the n-tuple $D^{(n)}$ denoting the product term which disappears from the Karnaugh map on a D-fault.

A growth (G), or shrinkage (S) cube is, respectively, a G or S term augmented (or concatenated) with the OCV of the corresponding source cube of the PLA. For example, if 1x01 011 is a cube (i.e., row) of a PLA (n=4, p=3), then, 0x01 011 is a G-cube, where 0x01 is the G-term; 1001 011 is an S-cube, where 1001 is the S-term.

A disappearance (D), or an appearance (A) cube is a D or A term augmented with an OCV denoting a propagated single fault. Thus, for the example input cube 1x01 011 the two possible D-cubes are: 1x01 001 and 1x01 010, whereas, the only possible A-cube is 1x01 100.

A minterm is any n-bit vector $(m_1m_2...m_n)$, $m_i \in [0,1]$, representing a single vertex of the n-dimensional (Boolean) hypercube one might consider in analyzing n-variable Boolean functions.

A mincube of a PLA is defined to be a cube consisting of a minterm as the input part and an OCV which defines the correct (fault-free) output values.

We use the symbols G, S, D and A to denote the sets of growth, shrinkage, disappearance and appearance faults, respectively, of a PLA.

IV. FAULT ANALYSIS AND TEST GENERATION

Algorithms to derive all G, S, D and A terms and/or cubes from the input PLA cubic specification are straightforward, and are presented in [5] in detail.

More Definitions

For a given output function, a minterm of a product term is said to be free if it is not covered by any other product term of the function under consideration.

A term $Y^{(n)}$ is said to subsume ($\subseteq$) another term $Z^{(n)}$ if every minterm covered by $Y^{(n)}$ is also covered by $Z^{(n)}$. Thus, for example, 110x $\subseteq$ 1x0x. Generalizing, a term $Y^{(n)}$ is said to subsume a set of terms $T_1, T_2, \ldots, T_k$ if every minterm covered by $Y^{(n)}$ is also covered by at least one of the terms T_i, $i = 1, 2, \ldots, k$.

Two OCVs are said to be incompatible if one is the bitwise complement of the other; otherwise, they are compatible. Two OCVs are said to be j-compatible, $1 \le j \le p$, if their jth bits are both 1.

An OCV O_1 is said to cover another OCV O_2 if O_1 has a '1' in every position in which O_2 has a '1'; we write, in this case, $O_1 \geqslant O_2$ (or, equivalently, $O_2 \leqslant O_1$.

Let C denote the set of m cubes, and P the set of m product terms of a (n,m,p)-PLA. Also, let G_c, S_c, D_c, and A_c denote, respectively, the set of G, S, D and A cubes of the PLA.

Undetectable faults

Due to the presence of redundant crosspoints in PLAs, there are usually quite a few undetectable faults; these could belong to any of the sets G, S, D and A. Before generating a single fault test set, it is often wise to remove undetectable faults from the set of modeled faults. In terms of our fault analysis and testing methods, this calls for removal of redundant cubes from the sets G_c, S_c, D_c, and A_c.

Lemma: A G-cube g is redundant, and can be removed from G_c if the following condition is satisfied:

$g_{i.p} \subseteq P_i$, for $i \in O_g$, where O_g = set of integers describing the output connections of g; the P_i's are subsets of P such that the OCVs corresponding to the product terms in P_i are i-compatible with $g_{o.p}$.

Corollary: A G-cube g is redundant and can be removed from G_c if the following condition is satisfied:

$g_{i.p} \subseteq c_{i.p}$ and $g_{o.p} \leqslant c_{o.p}$, for some $c \in C$.

Similar results depicting the conditions for redundant S, D and A cubes can be stated and proved [5]; these are omitted here for brevity.

G, S, D and A Tests

Clearly, any free minterm covered by a G-term qualifies as a test vector for the corresponding G-fault. For a given set of G-terms, a set of free minterms, chosen to cover all the G-terms is a (single) G-fault test set. Similarly, one can define S, D and A test sets.

The following lemmas form the basis of our test generation procedures, and are also useful in establishing bounds on test set size.

Lemma: The number of G-tests, n^g, needed for a product term t is bounded as follows:

$$0 \le n^g \le k,$$

where, k = number of literals (non-x elements) of t.

An efficient procedure (algorithm) to generate G-tests from G_c, called "the tabular covering method" has been described in [5]. A brief illustration of this procedure was presented earlier in [3].

Lemma: The number of S-tests, n^s, needed for a general product term t is bounded as follows:

$$n^s \le n - k + 1,$$

where, n = number of input variables, k = number of literals in t.

In [5], we have presented simple algorithms to generate a minimal S-test set for a (n,m,p)-PLA, using the "opposite-adjacent" method. We have also shown that in most cases, the upper bound on the number of S-tests for a product term of arbitrary size is usually only 2 or 3.

A simple example (Figure 1) is cited here to illustrate the method of generating a S-test set for a given product term, t. Let t = 1xxx. First choose a free minterm t_1 = 1100 (say). Choose a second minterm t_2 by complementing the x-values chosen for t_1. Thus, t_2 = 1011, which is "opposite" to t_1 within t. If t_2 is also free, then (t_1, t_2) forms the minimal S-test set for t (Figure 1a). If there is no free t_1 for which a free t_2 exists then upto two free minterms "adjacent" to t_2 (i.e., at unit binary distance from t_2) must be chosen to cover the remaining shrinkages. This is illustrated in Figure 1b, where shaded positions indicate bound minterms of our example product term 1xxx.

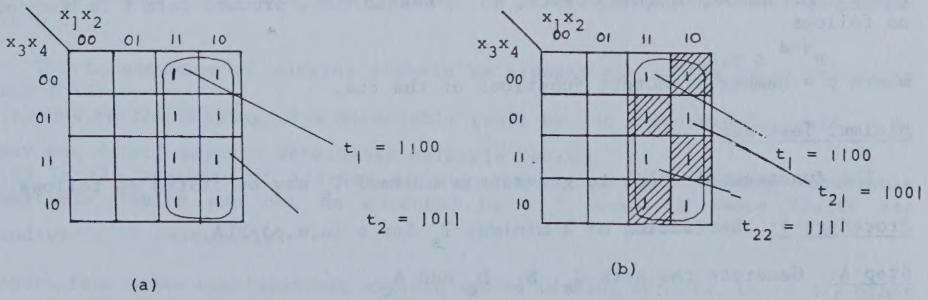


Figure 1. Illustration of "opposite-adjacent" method

If it is impossible to choose valid "adjacent" minterms as above, the "No-opposite-adjacent" (N-O-A) condition is said to exist; such a scenario requires further manipulation to derive the complete S-test set for the product term under consideration. However, as shown in [5], the N-O-A condition is extremely rare (if not totally non-existent) in any practical PLA implementation; hence, the S-test set of a product term is usually upper bounded by 3. The detailed "opposite-adjacent" method and algorithm is described in [5].

Let us now consider A- and D-tests. For the ith product term p_i, let q be the number of output functions to which p_i normally belongs. In other words, the OCV corresponding to p_i has q 1's, where, $1 \leq q \leq p$ (p is the number of output functions). If $q = p$, then no A-faults can be generated by p_i and hence no A-terms or A-cubes need be considered. If $q < p$, then clearly, $(p - q)$ A-cubes need to be considered. For example, if p_i = x10x1 0011, then the A-cubes to be generated and considered are: x10x1 1000 and x10x1 0100.

An A-fault, like a G-fault causes extra true minterms to appear. A logical way to handle A-faults is to treat the set of A-cubes, A_c, as a set of additional G-cubes. In other words, A_c may be merged with G_c before using the "tabular covering" method to generate a (G+A) test set.

However, a second viable strategy is to generate A-tests during the generation of S-tests. In fact, S, D, and A-tests can all be generated simultaneosly for a given product term in quite an efficient manner. Usually, an S-test set for a given product term (generated using the "opposite-adjacent" method, say), will automatically test for most (if not all) the D- and A-faults of that product term. However, in certain cases, additional test vectors are required to cover all disappearances and appearances.

In [5] we have presented a general algorithm to generate an (optionally) minimal S-D-A test set for a (n,m,p)-PLA, using an "augmented opposite-adjacent" method.

Lemma: The number of (D+A) tests, n^{d+a}, needed for a product term t is bounded as follows:

$$n^{d+a} \lesssim p,$$

where p = number of output functions of the PLA.

Minimal Test Set

The fundamental steps to generate a minimal T_s may be listed as follows:

Procedure 1: Derivation of a minimal T_s for a (n,m,p)-PLA.

Step A: Generate the sets G_c, S_c, D_c and A_c.

Step B: Eliminate redundant cubes to obtain the modified sets G_c', S_c', D_c' and A_c'.

Step C: Choose a set of mincubes to minimally cover the sets obtained in Step B.

The set in Step C forms a minimal T_s.

Any minimal covering problem is known to be NP-complete. Several faster, near-minimal test set generation procedures have been studied [5]. We present below one such procedure which has proven to be most efficient.

Procedure 2: Faster, near-minimal T_s generation.

Step A: Derive a minimal (S+D) test set T^{S+D}, using the "opposite-adjacent" method.

Step B: Derive a minimal (G+A) test set T^{G+A}, using the "tabular covering" method referred to earlier.

Then, $T_s = T^{S+D} \cup T^{G+A}$.

Bound on size of T_s

Theorem: For a general (n,m,p)-PLA, the size of a T_s covering all single G, S, D and A faults is bounded as follows:

$$|T_s|^{G+S+D+A} \leq m(n+p).$$

Thus, for a PLA with 16 input variables, 48 product terms and 8 output functions, $|T_s| < 1152$.

Multiple fault detection test set T_M.

As pointed out in [2,5,8], due to the presence of undetectable (single) faults in a PLA, and the phenomenon of fault masking, a single fault test set T_s is often inadequate in testing for multiple failures. The following two lemmas [6,8] express the fault masking conditions:

Lemma: A fault from the set (G U A) existing on some cube C_j of a PLA can be masked only by a fault from the set (S U D) also existing on the same cube C_j.

Lemma: A fault from the set (S U D) existing on some cube C_j of a PLA can only be masked by a fault from the set (G U A) existing on some cube other than C_j.

The consequence of masking effects as brought out by the above lemmas is two-fold:

(a) Due to the masking of a detectable fault by an undetectable fault, a T_s may not detect certain detectable multiple faults.

(b) In the presence of a 4-way masking phenomenon [6], certain detectable multiple faults may not be detected by a T_s even if those faults are individually detectable.

Apart from these complications arising out of masking effects, there are other cases [5] (e.g. a detectable G-G type fault in which each G fault is individually undetectable) where a T_s may not cover multiple faults.

The following procedure generates a multiple fault detection test set, T_M. For a proof/rationale and discussions of efficiency (in terms of speed and test set size, see [5]).

Procedure 3: Derivation of a T_M for a (n,m,p)-PLA.

Let C be the set of cubes of the PLA.

1. Generate G_c, S_c, D_c and A_c.

2. Eliminate redundant cubes to obtain the modified sets G_c', S_c', D_c' and A_c'.

In doing this step, the following book-keeping operations must be done:
(i) If there are two (or more) redundant G-cubes corresponding to a single cube in C, then a new cube is created by injecting any one of these undetectable G faults. This cube is then added to the original set C if it doesn't already exist there.
(ii) For every redundant S-cube which is eliminated, a new cube is created by injecting the corresponding S-fault into the source cube. This new cube is added to C if it doesn't already exist.
(iii) If a given cube in C has a redundant G-cube as well as a redundant A-cube, then this cube in C is modified by injecting the undetectable A-fault.

3. If a cube in C was added or modified, then goto step 1, else continue.

4. Choose a set of mincubes to cover the sets G_c', S_c', D_c' and A_c'. This set forms the required T_M.

(Note that instead of doing a full covering, Procedure 2 may be followed for efficiency once the set C has stabilized after steps 1-3 above).

V. DESIGN FOR TESTABILITY

Based on the insight gained from the above logical fault analysis methods, a simple procedure for designing PLAs for ease of test generation is briefly presented.

Definitions:

A product term t of a PLA is said to be totally S-observable (TSO), if every single S-fault in t is detectable, otherwise it is said to be "non-TSO."

Similar definitions are applicable for totally G-observable (TGO), totally D-observable (TDO) and totally A-observable (TAO).

Two product terms p_i and p_j are said to be mutually touching if there is at least one minterm which is covered by both p_i and p_j. Otherwise, they are said to be mutually non-touching.

Clearly, a PLA in which every product term is TGO, TSO, TDO and TAO is a crosspoint-irredundant PLA. A PLA in which every product term is TGO and TDO is called (G-D)-irredundant; similar definitions apply to other types of "partial" irredundancies.

As proved in [9], it is not possible, in general, to have a crosspoint-irredundant PLA realization of an arbitrary set of functions without introducing extra control inputs as in [8]. However, it *is* possible to implement any set of functions as a G-irredundant, S-irredundant, (S-D)-irredundant, (G-D)-irredundant, (S-A)-irredundant, (S-D-A)-irredundant, A-irredundant or (D-A)-irredundant PLA. The above types of partially irredundant realizations exhibit different characteristics and trade-offs in terms of testability and size. A detailed discussion of these trade-offs is presented in [9], along with specific procedures (algorithms) for generating such PLAs from the original product term specification. Following are some of the lemmas which form the basis for these procedures.

Lemma: A PLA realization in which the product terms of each output function are mutually non-touching, is (S-D)-irredundant.

(Such a PLA realization has the following desirable property: a 4-way masking effect cannot be present in such a PLA).

Lemma: A PLA realization in which all the product terms are mutually non-touching, is (S-D-A)-irredundant.

Lemma: A PLA in which each output function is realized as a minimal cover of prime implicants, is (G-D)-irredundant.

The following lemma expresses the upper bound on the size of $|T_s|$ for a (S-D-A)-irredundant PLA.

Lemma: For a (S-D-A)-irredundant (n,m,p)-PLA, the size of T_s can be shown to be bounded as follows:

$$|T_s| \leq m(n+1).$$

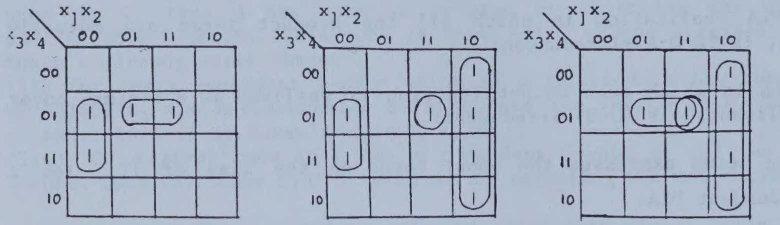
The test generation complexity for many of the partially irredundant PLAs is considerably lower than that for a general, redundant PLA. The following experimental results, shown in Table 1, (obtained by running commercial PLAs through our program PLATEST [5]), shows the changes in m, $|T_s|$ (i.e., size of T_s), and T (test generation time) for two different PLA realizations. (In each case, results for the modified PLAs are normalized with respect to those for the original, unmodified PLAs).

Table 1

Original PLA parameters	(S-D-A)-irredundant form			(G-D)-irredundant form		
	Δm (%)	$\Delta\|T_s\|$ (%)	ΔT (%)	Δm (%)	$\Delta\|T_s\|$ (%)	ΔT (%)
n=16, m=48, p=8	+6.25	-1.33	-22.4	-8.33	+3.33	+0.4
n=12, m=24, p=4	+8.33	-0.33	-19.6	-6.45	+5.0	-0.05
n=8, m=16, p=4	+10.44	-0.76	-20.6	-5.55	+4.25	+0.4

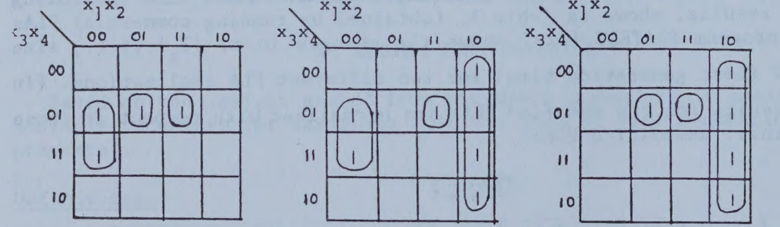
The results show that conversion to (G,D)-irredundancy often results in a reduction of m, and hence the PLA size, at the cost of slightly increasing T_s and without appreciable change in computational complexity. However, conversion to (S-D-A)-irredundant realization results in considerably more efficient test generation, at the cost of a slight increase in PLA size, and without appreciable effect on test set size. More exhaustive experimentation with a large number of PLAs is currently under way. On the basis of our initial set of experiments, we have come up with a systematic methodology for PLA implementation of a given set of functions, under various constraints of test set size, multiple fault coverage and PLA size. This is reported in a forthcoming, detailed report [9].

We conclude, by citing a simple example of easily testable PLA design. Figure 2a shows the unmodified PLA example cited in Figure 1 of [8]. In that paper, the authors have described a method of adding extra control inputs to the PLA, to make it crosspoint irredundant. Figure 2b shows an S-D-A irredundant realization of the same PLA, which does not require an increase in any of the parameters n, m or p. Similarly, Figure 2c shows a G-S-D irredundant realization. Such designs ensure (i) a small T_s, (ii) an easily derivable T_s and (iii) complete multiple fault coverage by T_s.



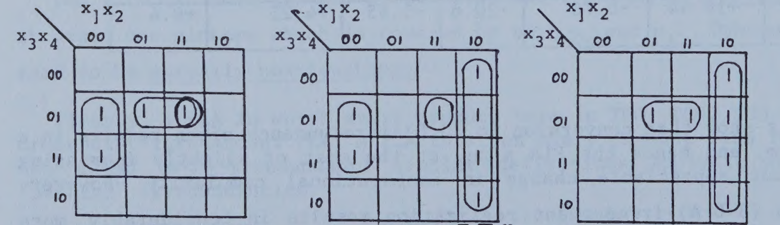
$f_1 = x_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_4$ $f_2 = x_1\bar{x}_2 + \bar{x}_1\bar{x}_2x_4 + x_1x_2\bar{x}_3x_4$ $f_3 = x_1\bar{x}_2 + x_2\bar{x}_3x_4 + x_1x_2\bar{x}_3x_4$

(a) Original PLA: n=4, m=4, p=3



$f_1 = \bar{x}_1\bar{x}_2x_4 + \bar{x}_1x_2\bar{x}_3x_4 + x_1x_2\bar{x}_3x_4$ $f_2 = x_1\bar{x}_2 + \bar{x}_1\bar{x}_2x_4 + x_1x_2\bar{x}_3x_4$ $f_3 = x_1\bar{x}_2 + x_1x_2\bar{x}_3x_4 + \bar{x}_1x_2\bar{x}_3x_4$

(b) S-D-A irredundant PLA: n=4, m=4, p=3



$f_1 = x_2\bar{x}_3x_4 + \bar{x}_1\bar{x}_2x_4 + x_1x_2\bar{x}_3x_4$ $f_2 = x_1\bar{x}_2 + \bar{x}_1\bar{x}_2x_4 + x_1x_2\bar{x}_3x_4$ $f_3 = x_1\bar{x}_2 + x_2\bar{x}_3x_4$

(c) G-S-D irredundant PLA: n=4, m=4, p=3

Figure 2. Transformation to partially irredundant PLAs

REFERENCES

[1] D. L. Ostapko and S. J. Hong, "Fault analysis and test generation for programmable logic arrays," IEEE Trans. Comput., vol. C-28, pp. 617-626, Sept. 1979.

[2] J. E. Smith, "Detection of faults in programmable logic arrays," IEEE Trans. Comput., vol. C-28, pp. 845-853, Nov. 1979.

[3] P. Bose and J. A. Abraham, "Test generation for programmable logic arrays," Proc. 19th Design Automation Conf., Las vegas, June 1982, pp. 574-580.

[4] P. Bose, "Generation of minimal and near-minimal test sets for programmable logic arrays," Proc. Int'l. Conf. on Computers, Systems and Signal Processing, Bangalore, India, December 1984, pp. 702-706.

[5] P. Bose, "Functional testing of programmable logic arrays," IBM Tech. Rept. #RC 10681, Yorktown Heights, NY, 1984.

[6] V. K. Agarwal, "Multiple fault detection in programmable logic arrays," IEEE Trans. Comput., Vol. C-29, pp. 518-522, June 1980.

[7] S. M. Reddy and D. S. Ha, "On the design of testable PLAs," Proc. 1985 Conf. on Info. and Systems & Sciences, Johns Hopkins Univ., March 1985.

[8] K. S. Ramanatha and N. N. Biswas, "A design for testability of undetectable crosspoint faults in programmable logic arrays," IEEE Trans. Comput., vol. C-32, pp. 551-557, June 1983.

[9] P. Bose, "Designing PLAs for ease of test generation," IBM Tech. Rept., Yorktown Heights (in preparation).

VECTOR SETS FOR VLSI TESTING

Gadiel Seroussi and *Nader H. Bshouty*
Technion, Israel Institute of Technology
Haifa 32000, Israel

1. Introduction

Consider a binary logic circuit C with L inputs $R_1, R_2, \cdots, R_L$, and W outputs $O_1, O_2, \cdots, O_W$, such that each output depends on (at most) d inputs.[1] Let T be a subset of the L-dimensional space $\{0,1\}^L$, and let $t = |T|$ denote the cardinality of T. We shall also regard T as a $t \times L$ binary matrix, whose rows are the vectors of T, arranged in some arbitrary order. Let $I = \{i_1, i_2, \cdots, i_d\}$ be a set of d input positions (i.e. $1 \le i_j \le L$ for $1 \le j \le d$). The set $R_I = \{ R_i \mid i \in I \}$ is called an *input set*. The *projection* T_I of T onto I is the $t \times d$ matrix

$$T_I = [T_{i_1} T_{i_2} \cdots T_{i_d}],$$

where $T_1, T_2, \cdots, T_L$ denote the columns of T. We say that T *exhaustively covers* the input set R_I if the rows of T_I include all 2^d binary d-tuples. T is an *exhaustive test set* for C if for each output O_j of C, such that O_j depends on the input set R_{I_j}, T exhaustively covers R_{I_j}. T is an *(L,d)-universal test set* if it exhaustively covers all $\binom{L}{d}$ input sets. Clearly, an (L,d)-universal test set is an exhaustive test set for any circuit C with the given parameters L and d. Exhaustive test sets are useful for testing VLSI circuits with a large number of functional components, designed such that each component depends on a limited number of inputs. The test set T is run on the inputs, and the resulting output sequences (or, rather, their compressed *signatures*) are compared to pre-computed correct values. In a special case of particular interest, the inputs are arranged on a shift-register R of length L, and the test set T consists of the states of R when a *test sequence* S is fed into one end of the register, and linearly shifted through it. We call this *sequential testing*, as opposed to *parallel testing*, where each vector in the test set T is fed in parallel into the inputs of the circuit. We call the cardinality of T, or the length of S, the *size* of the test. Since the the time consumed by the test is proportional to the size, our objective is to design test sets or sequences of size as small as possible. Another important parameter is the amount of memory needed to generate the test set or sequence. In the worst case, the whole test set or sequence must be stored. Several constructions of test sets and sequences can be found in the literature. Examples of recent papers are [1], [2], and [3] for parallel testing, and [4], [5], and [6] for sequential testing. In this paper, we present a probabilistic construction of universal test sets, based on the properties of maximal length shift register sequences. The result of the construction might fail to be an (L,d)-universal test set, with a controllable probability which decreases exponentially with the size of the test. For practical applications, we give a simple checking procedure which enables us to construct test sets where the probability of coverage failure is zero. Our test sets are either smaller than those obtained from previous constructions, or, for the same size, require substantially less memory.

---

[1] We can assume, without loss of generality, that each output depends on exactly d inputs.

2. Test sets

Consider an array of L linear feedback shift registers (LFSRs) of length d, all of them wired according the coefficients of a fixed primitive polynomial of degree d over the binary field $GF(2)$. When such a register is initialized with a nonzero seed, it generates a binary sequence of maximal period 2^d-1 (this is called an *M-sequence*) [7]. Let $\mathbf{s} = [s_1 \, s_2 \, \cdots \, s_L]$ denote a $d \times L$ matrix whose columns s_i are used as seeds for the L LFSRs, and let $T_{\mathbf{s}} \subseteq \{0,1\}^L$ denote the set consisting of the all-zero L-tuple, and the 2^d-1 vectors obtained at the outputs of the LFSRs, when these are initialized with the seeds s_i, $1 \le i \le L$, and run for their whole period. The following proposition is readily derived from well known properties of M-sequences [7]:

Theorem 1: Let $I = \{i_1, i_2, \cdots, i_d\}$ be an arbitrary set of input positions. Then, $T_{\mathbf{s}}$ exhaustively covers R_I if and only if the $d \times d$ matrix $\mathbf{s}_I = [\, s_{i_1} \, s_{i_2} \, \cdots \, s_{i_d} \,]$ is non-singular over GF(2).

Assume $\mathbf{s}$ is chosen at random, with uniform probability. Then the probability of $T_{\mathbf{s}}$ covering R_I is equal to the probability of a uniformly distributed random binary matrix of order d being non-singular. This probability is equal to $\delta(d) = \prod_{i=1}^{d}(1-2^{-i})$ [8, p. 399]. It can be readily verified that for all $d \geq 1$,

$$\delta(d) > \delta = \lim_{k \to \infty}\delta(k) \approx 0.288.$$

Let $\mathbf{s}(1), \mathbf{s}(2), \cdots, \mathbf{s}(N)$ be N randomly chosen values of $\mathbf{s}$, and let

$$T = T_{\mathbf{s}(1)} \cup T_{\mathbf{s}(2)} \cup \cdots \cup T_{\mathbf{s}(N)}. \tag{1}$$

Then, the probability of T not covering an arbitrary input set R_I is upperbounded by $(1-\delta)^N$, and the probability of T being an (L,d)-universal test set is lowerbounded by $1 - \binom{L}{d}(1-\delta)^N$. Define[2] $\lambda = -1/\log(1-\delta) \approx 2.03$. Then, the above discussion leads to the following results:

Theorem 2: Let $N = \lceil \lambda d \log L \rceil$. Then, there exist N seed matrices $\mathbf{s}(1), \mathbf{s}(2), \cdots, \mathbf{s}(N)$ such that the set T as defined in (1) is an (L,d)-universal test set. The size of T is $O(d2^d \log L)$.

Theorem 3: Let $0 < \varepsilon < 1$, and let

$$N = \left\lceil \lambda \cdot (d \log L + \log \varepsilon^{-1}) \right\rceil.$$

Then, T as defined in (1) is an (L,d)-universal test set with probability greater than $1-\varepsilon$. The size of T is $O((d \log L + \log \varepsilon^{-1})2^d)$.

In a practical testing situation, we usually require the test to be reproducible so that signatures can be obtained from the output sequences and compared to pre-computed values. This requires the seed matrices $\mathbf{s}(1), \cdots, \mathbf{s}(N)$ to be randomly selected once, stored, and then used for the computation of the master signatures, and for the testing procedure. This requires $dLN = O(dL(d \log L + \log \varepsilon^{-1}))$ bits of memory for an (L,d)-universal test set. On the other hand, in practical applications, the number of outputs, and, hence, the number W of input sets to be covered, is much smaller than $\binom{L}{d}$. This, together with Theorem 1, enables us to construct a test set which exhaustively covers all required input sets with certainty. The construction proceeds as follows: randomly select seed matrices $\mathbf{s}(i)$, $i=1,2,3,\ldots$. For each seed matrix $\mathbf{s}(i)$, and for each yet uncovered input set R_I, check whether the $d \times d$ matrix $\mathbf{s}(i)_I$ is nonsingular. If so, mark R_I as "covered". Proceed until all required W input sets are covered. The expected number of seed matrices is $O(\log W)$, and the expected test size is $O(2^d \log W)$.

---

[2] All logarithms are to base 2.

The existence of (L,d)-universal test sets of size $O(d \log L 2^d)$, and the fact that a random test set of size $c \cdot d \log L 2^d$ is with high probability an (L,d)-universal test set, were also proved in [1]. There, however, each test vector is chosen independently at random, and hence, the whole test set has to be stored. It is interesting to notice that we obtain the same test size even when imposing on the test set the algebraic structure of a union of vector spaces of dimension at most d. This structure allows us to generate the test set from a much smaller amount of stored data, and also provides an efficient way of checking individual input sets for coverage. This is not possible in unstructured constructions.

The following theorem shows that for the particular algebraic structure imposed on T, the results of Theorems 2 and 3 are not very far from optimal.

Theorem 4: Let

$$T = T(1) \cup T(2) \cup \cdots \cup T(N),$$

where $T(i)$, $i \le i \le N$, are subspaces of dimension d of $\{0,1\}^L$. Assume that T is an (L,d)-universal test set such that every input set R_I is covered by at least one of the subspaces $T(i)$. Then, $|T| \ge \Omega(2^d \log L)$.

Other deterministic constructions in the literature give tests of much larger sizes than those given in Theorems 2 and 3. The deterministic construction in [1] gives a test set size of $O(2^d (\log L)^{d-1})$. The construction in [3] gives a test set of size $O(g(d)(\log L)^{\alpha(d)})$, where $g(d) > 2^d$, and $\alpha(d) \approx \log d^2 / 4$. In both cases, the whole test set must be stored, and, hence, the memory requirement is L times the size of the set. A probabilistic construction of test sequences, also based on M-sequences, can be found in [6], where a size of $O((d\log L+\log\varepsilon^{-1})2^{M(d+\lceil\log_e(L-d)\rceil)})$ is obtained. Here $M(k)$ denotes the smallest prime greater than or equal to k, such that $2^{M(k)}-1$ is a prime, e is the basis of natural logarithms, and ε is the probability of the constructed sequence failing to be (L,d)-universal. For favorable values of L and d, such that $M(d+\log_e(L-d)) \approx d+\log_e(L-d)$, and for fixed ε, this gives a test size of $O(d2^d(L-d)\log L)$. In general, however, the values of $M(k)$ are unknown for all but a finite number of positive integers k.

We conclude with a comparative table of the different constructions, for typical values of L and d. *Size* denotes the size of the test, *Mem* denotes the amount of memory required (in bits), and P_{fail} denotes the probability of the test failing to be (L,d)-universal.

Source	$L=500$, $d=10$ *Size*	*Mem*	$L=500$, $d=20$ *Size*	*Mem*	P_{fail}
Theorem 3	$4\cdot 10^5$	$2\cdot 10^6$	$6\cdot 10^8$	$6\cdot 10^6$	2^{-100}
[6]	$2\cdot 10^{11}$	$4\cdot 10^3$	$3\cdot 10^{11}$	$5\cdot 10^3$	2^{-100}
[3]	$3\cdot 10^6$	$2\cdot 10^9$	$3\cdot 10^{15}$	$2\cdot 10^{18}$	0
Deterministic [1]	$4\cdot 10^{11}$	$2\cdot 10^{14}$	$1\cdot 10^{24}$	$5\cdot 10^{28}$	0
Random set [1]	$1\cdot 10^5$	$6\cdot 10^7$	$2\cdot 10^8$	$1\cdot 10^{11}$	2^{-100}

REFERENCES

[1] A.K. Chandra, L.T. Kou, G. Markowsky, and S. Zaks, "On sets of Boolean n-vectors with all k-projections surjective", *Acta Informatica*, Vol. 20, pp. 103-111, 1983.

[2] D.T. Tang and Lin S. Woo, "Exhaustive test pattern generation with constant weight vectors", *IEEE Trans. Comput.*, Vol. C-32, pp. 1145-1150, 1983.

[3] D.T. Tang and C.L. Chen, "Iterative exhaustive pattern generation for logic testing", *IBM J. Res. Develop.*, Vol. 28, pp. 212-219, 1984.

[4] Z. Barzilai, D. Coppersmith, and A.L. Rosenberg, "Exhaustive generation of bit patterns with applications to VLSI self-testing", *IEEE Trans. Comput.*, Vol. C-32, pp. 190-194, 1983.

[5] D.T. Tang and C. Chen, "Logic test pattern generation using linear codes", *IEEE Trans. Comput.*, Vol. C-33, pp. 845-849, 1984.

[6] A. Lempel and M. Cohn, "Design of universal test sequences for VLSI", *IEEE Trans. Info. Theory*, Vol. IT-31, pp. 10-17, 1985.

[7] S.W. Golomb, *Shift Register Sequences*, San Francisco: Holden Day, 1967.

[8] F.J. MacWilliams and N.J.A. Sloane, *The Theory of Error Correcting Codes*, Amsterdam: North-Holland, 1978.

On the Generation of Test Vectors for Multiple Faults in Digital Circuits

SREEJIT CHAKRAVARTY and H.B.HUNT III [1]
Dept. Of Computer Science
State University of New York
1400 Washington Ave.
Albany , NY 12222.

ABSTRACT

The problem of generating test vectors for multiple faults in combinational circuits is considered. Stuck-at and functional_gate faults[3] are considered. We show that generating a test vector for a multiple stuck-at fault is no harder than generating a test vector for a single stuck-at fault. This is in contrast to the performance of the D-Algorithm which is exponential in the multiplicity of the fault.

We propose a new algorithm which does not have this bad property. The complexity of our algorithm is $O(4^f m)$ where f is the number of head fanout points and m the number of gates in the circuit. In [2] the same complexity bound was obtained for single stuck-at faults. We also compare the potential usefulness of our algorithm vis-a-vis the D-Algorithm[1] and PODEM[4].

1. Introduction

We consider the problem of generating test vectors for multiple faults in combinational circuits. We assume multiple occurrences of stuck-at and functional_gate_faults. (A Fault in which the function of a gate has changed from g to h will be called a functional_gate_fault(g,h)[3]. If h is 1[or 0] then we have a stuck-at-1[or 0] fault.)

In sections 2 and 3 we present an algorithm for this problem. In section 4 we prove that generating a test vector for such a fault is no harder than generating a test vector for a single stuck-at fault. Existing algorithms, *viz* the D-Algorithm can be exponential in the multiplicity of the fault. The potential usefulness of our algorithm is discussed in section 5.

2. Preliminaries

We use the logic values $\{0,1,d,\overline{d}\}$[1]. At the location of s-a-1, s-a-0 and functional_gate_faults(g,h) we introduce the **pseudo gates** α, β and γ(g,h) respectively. The circuit with the **pseudo gates** introduced is called the **augmented circuit**. Introduction of the **pseudo gates** eliminates the need for fault cubes[1] and leads to an improved search strategy. Propagation cubes for the pseudo gates α and

---

[1] This research was supported by NSF Grant No. DCR84-03014.

β are shown in table 1. Propagation cubes for $\gamma(g,h)$ can be obtained from the truth tables of g and h[1].

Table 1

Input = x,
Output = y.

Gate	x	y
α	0	d
	1	1
	d	d
	$\bar{d}$	1
β	0	0
	1	$\bar{d}$
	d	0
	$\bar{d}$	$\bar{d}$

Figure 1. Propagation D_Cubes for the α and β gates.

A point in the augmented circuit, for an input assignment, has the value 0(has the value 1) if the corresponding points in the faulty and fault free circuits has the value 0(has the value 1,respectively); it has the value d(has the value $\bar{d}$) if the corresponding points in the faulty and fault free circuits has the value 1 and 0 (0 and 1), respectively.

A fanout point p is said to be a **head fanout point** (abbreviated **HFP** [2]) iff there exists a path from a primary input to p such that the path does not pass through any other fanout point. All other fanout points are said to be **dependent fanout points**. In figure 5, A,B,C are HFPs and D is a dependent fanout point.

Notation:

Y -- primary output
f -- number of head fanout points
n -- number of primary inputs
m -- number of gates in the augmented circuit
$X = \{ x_1, ..., x_n \}$ -- set of primary inputs
G -- an arbitrary gate with inputs $g_1, ..., g_t$ and output g
$F = \{ F_1, ..., F_f \}$ -- set of HFPs
$A = (a_1, ..., a_f), a_i \in \{ 0,1,d,\bar{d} \}$ -- an assignment of values to the HFPs;
Value of $F_i = a_i$.

The following two easy propositions are necessary to understand the algorithm. In the sequel we make a distinction between a fanout point and the fanout branches of the fanout point.

Proposition 2.1 The number of HFPs is less than or equal to the number of primary inputs.

Proposition 2.2 An assignment of values to all the HFPs determines the values of all the fanout points.

An assignment of values to the HFPs is said to be **consistent** iff there exists an input vector v such that the values at the HFPs for the input vector v are the values dictated by the assignment. All other assignments to the HFPs are said to be **inconsistent assignments**.

3. Description of the algorithm

The algorithm enumerates the assignments to the head fanout points. For each assignment of values to the HFPs, procedure FIND_VECTOR (figure 2) is invoked. If for any of these assignments FIND_VECTOR succeeds in finding a test vector, the algorithm terminates. If for every assignment FIND_VECTOR fails to find a test vector, the fault is not testable.

```
Procedure FIND_VECTOR
    (* Let $A = < a_1, ..., a_f >$ be the given assignment. *)
    $PS(x_i) = \{0,1\}$ , $\forall i = 1,2,..,n$
    Gate_no = 1 ; Consis = TRUE ;
    WHILE ((Gate_no $\leq$ m) AND ( Consis )) DO
     BEGIN
      (* Let G be the gate, g the output and $g_1,.., g_t$ the inputs to G *)
      Compute PS(g) from $PS(g_1),...,PS(g_k)$ ;
      IF ( g is a head fanout point ) THEN
       BEGIN
          (* check if the assignment is consistent or not *)
         Consis = constant value assigned to g $\in$ PS(g);
          If( NOT Consis ) then return;
         PS(g) = { constant value assigned to g in A };
          (* Let $b_1 ,.., b_t$ be the fanout branches of g. *)
         $\forall i$ ,PS( $b_i$ ) = PS(g);
       END;
      Gate_no = Gate_no + 1;
     END;
     (* backward sweep *)
    IF (Consis AND ( d or $\bar{d} \in$ PS(Y))) THEN
     BEGIN
      Mark Y with d or $\bar{d} \in$ PS(Y);
      FOR Gate_no m DOWNTO 1 DO
       Deduce an assignment to the inputs of the gate ;
       (* This can be done using the propagation d-cubes *)
      The marking on the primary inputs give the test vector;
     END
    ELSE
     The given assignment does not work ;
    RETURN;
```

Figure 2

Given an assignment A, FIND_VECTOR associates with all points W in the augmented circuit a set $PS(W) \subseteq \{0,1,d,\bar{d}\}$. PS(W) is the set of possible values that W can have if the HFPs are assigned A. It initializes the sets $PS(x_i)$, $1 \leq i \leq n$ to $\{0,1\}$. It makes a forward sweep until it detects the assignment to be inconsistent or computes PS(Y). If the assignment is inconsistent or d or $\bar{d} \notin PS(Y)$ FIND_VECTOR fails to find a test vector for the given assignment.

Let G be a gate, g its output and $g_1, \ldots, g_t$ its inputs. To compute PS(g) for a gate G, compute $W' = PS(g_1) \times \cdots \times PS(g_t)$. For all $w \in W'$ find w' s.t. the gate G outputs w' if the inputs are assigned w, using the propagation d-cubes. $PS(g) = \bigcup_{w \in W'} \{w'\}$.

If g is a HFP, let $b_1, \ldots, b_p$ be its fanout branches. Let a be the value dictated by the given assignment. The assignment is consistent iff $a \in PS(g)$. $PS(b_i) = \{ a \}$.

If the algorithm succeeds in the forward sweep, it makes a backward sweep. The output is marked by d or $\bar{d} \in PS(Y)$. Every other point x in the circuit is marked by an element $q \in PS(x)$, computed in the previous phase, using the propagation d-cubes. The markings on the primary input gives the test vector.

4. Complexity results

A folk theorem suggests that it is harder to generate a test vector for a multiple stuck-at fault than for a single stuck-at fault. This belief stems from the performance of the D-Algorithm. The D-algorithm is exponential in the multiplicity of the fault assumed. However, the next theorem suggests that an algorithm for generating a test vector for a multiple fault need not have this bad property.

Theorem 4.1. The problem of generating a test vector for a multiple stuck-at fault is linearly reducible to the problem of generating a test vector for a single stuck-at fault.

Proof:- Let C be the circuit under consideration and F the fault under consideration. Let F be of multiplicity q, and let $p_1, \ldots, p_q$ be the faulty locations s.t. p_i is stuck-at a_i, $1 \leq i \leq q$ and $a_i \in \{ 0,1 \}$.

From C and F we form the circuit C_F as follows. Cut the circuit C at the points $p_1, \ldots, p_q$ and let the points p_i be fixed at the constant value a_i, $1 \leq i \leq n$. The circuit so obtained is C_F. Let the outputs of C and C_F be **a** and **b** respectively. Form the circuit C_M with output **f** by making **a** and **b** the inputs to an XOR-gate. A test vector detects the fault F in C iff it detects the fault **f** stuck-at-0 in C_M. This concludes the proof. □

Theorem 4.2 follows from the correctness (see appendix) of our algorithm and the complexity of FIND_VECTOR.

Theorem 4.2. The complexity of finding a test vector for multiple faults in combinational circuits is $O(4^f m)$, where f is the number of HFPs and m is the number of gates in the augmented circuit.

Theorem 4.2 extends the result in [2] for single stuck-at faults. A direct extension of their method for multiple stuck-at faults is exponential both in the number of HFPs and the multiplicity of the fault and does not handle functional gate faults. Our algorithm behaves no poorer for multiple faults than the algorithm in [2] behaves for single stuck-at faults.

5. D_Algorithm and PODEM

In this section we discuss the potential usefulness of our algorithm. Consider the circuit of figure 3[4]. D-Algorithm performs very poorly for the fault marked[4]. PODEM performs well for this fault, so does our algorithm. The HFPs for this circuit are A,B,C and D. Since the fault does not flow through any of the HFPs, we will assign values to the HFPs from { 0,1 }. FIND_VECTOR will succeed for any of the assignments to the HFPs, *viz* (0,0,0,0), (0,0,0,1), ... (1,1,1,1). So, the first call to FIND_VECTOR succeeds.

The circuit in figure 4 is a modification of the circuit in Figure 3. Both the D-Algorithm and PODEM perform poorly for the fault marked. This is because PODEM has to go through a large number of input assignments before it can find one that sensitizes the fault. D-Algorithm performs poorly for the same reason it performs poorly on the circuit of figure 3. Our algorithm performs well for the same reason it performs well on the fault of figure 3.

The circuit of figure 5 is a modification of the circuit of figure 8 in [1]. For the fault marked in figure 8 of [1], multiple D-Paths are necessary. For this reason the D-Algorithm performs poorly on this circuit. PODEM too performs poorly. However our algorithm again performs well.

Acknowledgement: We thank S.S. Ravi for useful suggestions.

6. References

(1) J.P.Roth et-al : "Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits"; IEEE-TC, Vol EC-16, pp 567-580, Oct 1967.

(2) Hideo Toida, Shunichi Fujiwara : " The Complexity of Fault Detection problems for Combinational logic Circuits"; IEEE-TC, Vol C-31, No 6, 1982, pp 555-560.

(3) Airapetian and McDonald : "Improved test set generation for combinational circuit control"; 9th FTCS, 1978; pp 133-136.

(4) Prabhakar Goel, " An implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", IEEE-TC, Vol C-30, No 3, March 1981, pp 215-222.

APPENDIX : Sketch of proof of correctness.

We assume that all gates have bounded fanin. PS(x) denotes the set of possible values that the point x can take. The proof follows from the following three lemmas.

Lemma 1. Let T be a fanout free circuit with m gates whose output is Y. Let some of its inputs be assigned values from $\{ 0,1,d, \bar{d} \}$. Then,

(a) PS(Y) can be computed in time O(m).

(b) Given $b \in PS(Y)$, an assignment to the unassigned inputs which realizes the value b at Y can be computed in time O(m).

Proof. The proof is by induction on the number of gates in T. For $m=1$ the lemma is trivially true.

Suppose the lemma is true for all $m \leq q$. Consider a circuit with $q+1$ gates. Let G be the output gate of T, and $v_1, \ldots, v_p$ the inputs to G.

(a) Consider the subtrees of T, *i.e* T_i's whose output is v_i, $1 \leq i \leq p$. By the induction hypothesis $PS(v_i)$ can be computed in time $O(m_i)$, where m_i is the number of gates in T_i. Each of the subtrees T_i are mutually disjoint. Therefore, PS(Y) can be computed in time $O(\sum_{i=1}^{p} m_i) + O(1) = O(m)$.

(b) Let $b \in PS(Y)$ be given. The value of $v_i \in PS(v_i)$ i.e. a_i s.t. the gate G outputs b can be realized in constant time. In order to solve the original problem we solve for assignments to the inputs of T_i's for the value a_i at v_i. The proof follows from the induction hypothesis and the observation that the T_i's do not share any variable. ☐

Definition:- Let $R_1, \ldots, R_q$ be the fanout points of the circuit. Let $C_1, \ldots, C_q$ be the fanout free subcircuits with outputs $R_1, \ldots, R_q$ respectively. We define a partial order $\leq$ on the set of the fanout free subcircuits as follows: $C_i \leq C_j$ iff there exists a path from R_i to R_j.

Lemma 2. Given a circuit with m gates and an assignment to the HFPs the values at the dependent fanout points and the sets $PS(F_i)$, where the F_i's are the head fanout points, can be computed in time O(m).

Proof:- Consider the partial order of the fanout points defined above. Define levels in it as follows. Its minimal elements have level 1. For all other R_j, $\text{level}(R_j) = \max_{R_i \leq R_j} \{\text{level}(R_i)\} + 1$.

Suppose R_i is an element of level 1. Let C_i be the fanout free subcircuit with output R_i. All the inputs to C_i are primary inputs. Use Lemma 1 to compute $PS(R_i)$.

Proceed level by level. Let R_i be an element in the current level which is the output of the subcircuit C_i. Let $R_{i_1}, \ldots R_{i_p}$ be the HFPs whose fanout branches are the input to C_i. Let a_{i_j} be the value of the HFP dictated by the given assignment. Assign the value a_{i_j} to those inputs to C_i which are branches of R_{i_j}.

Let Q_{i_j}, $1 \leq j \leq t$ be the dependent fanout points whose branches are the inputs to C_i. Since level(Q_{i_j}) $<$ level(R_i) the value at Q_{i_j} is known. Let this value be b_{i_j}. Assign b_{i_j} to those inputs of C_i which are fanout branches of Q_{i_j}. Use Lemma 1 to compute $PS(R_i)$.

Lemma 1 is used exactly once for each of the fanout free subcircuits. $\square$

Lemma 3. Given a circuit with m gates and an assignment to the HFPs, the following can be done in time $O(m)$.

(a) Find if the assignment is consistent.

(b) Find if there exists a test vector for a given assignment, if the assignment is consistent.

(c) Find a test vector for a given assignment, if the assignment is consistent and a test vector exists for the assignment.

Proof. Let $F_1, \ldots, F_f$ be the HFPs in the circuit. Let $< a_1, \ldots, a_f >$ be the given assignment.

(a) From lemma 2 we can compute $PS(F_i)$ in time $O(m)$. The assignment is consistent iff $a_i \in PS(F_i)$, for all $1 \leq i \leq f$.

(b) If during the process of part (a) an inconsistency is not detected then the values of all the dependent fanout points are computed. Take the fanout free subcircuit whose output is the primary output of the circuit. Compute $PS(Y)$ using lemma 2. A test vector exists iff d or $\overline{d} \in PS(Y)$.

(c) To find an input assignment do the following. If the fanout free subcircuit whose output is the primary output of the circuit has a primary input as one of its input then mark the output by d or $\overline{d}$ whichever belongs to $PS(Y)$. Deduce the values of the primary inputs that are inputs to this fanout free circuit using lemma 1. For all the fanout free circuits whose outputs are the HFPs solve the same problem by marking the output by the value dictated by the given assignment. The markings on the primary inputs will give the test vector. $\square$

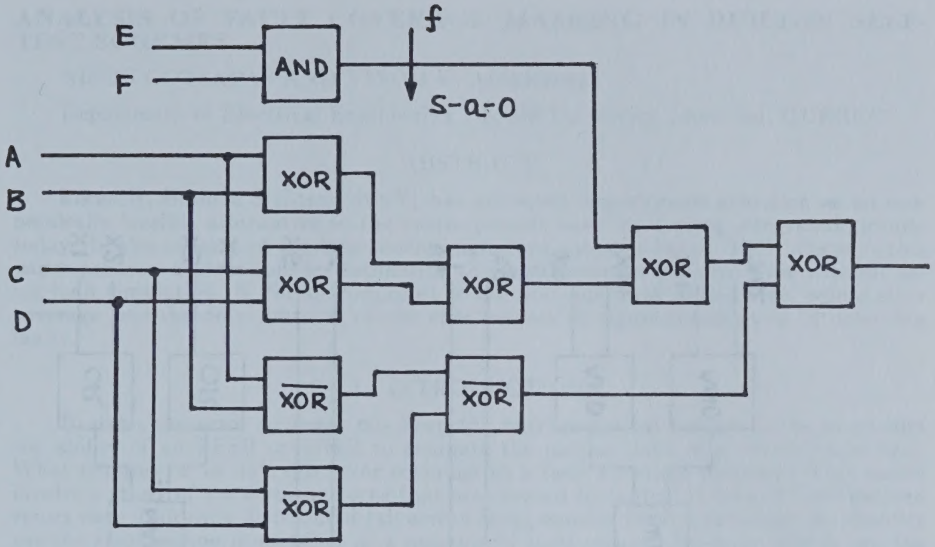

f
E
F
AND
S-a-0
A
B
C
D
XOR
XOR
XOR
$\overline{\text{XOR}}$
$\overline{\text{XOR}}$
$\overline{\text{XOR}}$
XOR
XOR

FIGURE -3

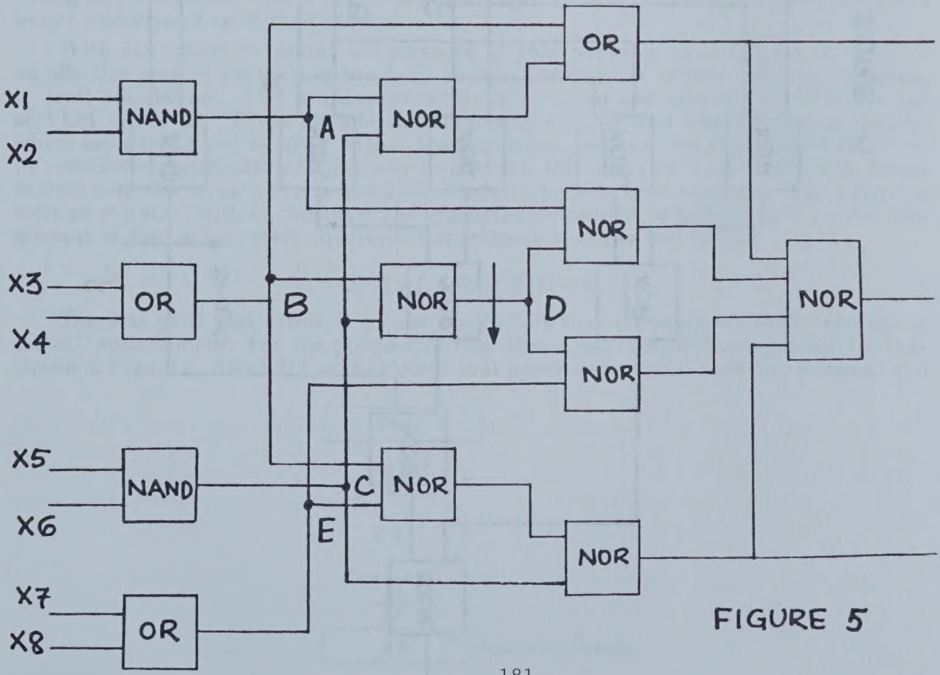

OR
X1
X2
NAND
A
NOR
NOR
X3
X4
OR
B
NOR
D
NOR
NOR
X5
X6
NAND
C
NOR
E
NOR
X7
X8
OR

FIGURE 5

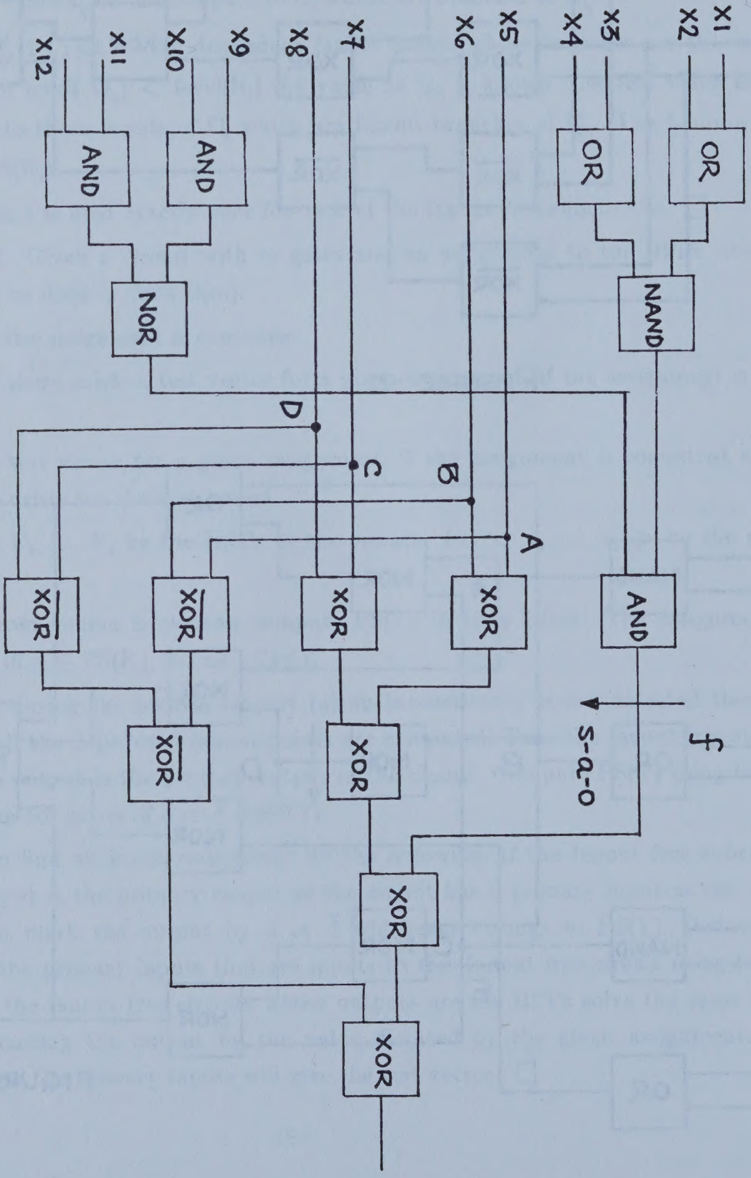

X1
X2
X3
X4
X5
X6
X7
X8
X9
X10
X11
X12
OR
OR
NAND
AND
AND
NOR
AND
A
B
C
D
XOR
XOR
$\overline{\text{XOR}}$
$\overline{\text{XOR}}$
XOR
$\overline{\text{XOR}}$
XOR
XOR
f
s-a-0

FIGURE 4

ANALYSIS OF FAULT COVERAGE MASKING IN BUILT-IN SELF-TEST SCHEMES

NICOS COTSAPAS AND VINOD K. AGARWAL

Department of Electrical Engineering, McGill University, Montréal, QUEBEC

ABSTRACT

Recently, Built-in Self-test (BIST) has attracted considerable attention as an economically feasible alternative to the costly process used for testing integrated circuits today. In the context of random testing/signature analysis-based BIST schemes, this paper presents a procedure for estimating the fault coverage in linear time without using fault simulation. It differs from most of the previous work in this area, where error coverage was the only measure of the effectiveness of signature analysis in detecting faults.

1. INTRODUCTION

To date, the error coverage has been the only analytical measure used to predict the ability of an LFSR or MISR to evaluate the output data of a circuit-under-test. What remains, is to link this error coverage to a fault coverage measure. This would involve a study of the error patterns that are created by faults. Indeed, if fault derived errors were uniformly distributed (all errors being equally likely), then one can directly use the error escape probability as a measure of fault escape. However, this is not the case, and a more realistic solution should be sought. To this end, Smith ([6]) made an attempt to classify certain types of fault-derived errors (such as burst errors and periodic errors) and to analyze the effectiveness of an LFSR in detecting these errors. While such measures can guide in the choice of the feedback polynomial of the LFSR, their validity is limited to certain classes of faults, and all potential errors are not included. Citing this limitation, Smith concludes that experimental research is needed to have a better understanding of fault classes.

With the above in mind, the purpose of this paper is to determine if there is an effective way of estimating the fault coverage in a BIST scheme without resorting to fault simulation. The primary intention is to tailor the analysis to a large class of VLSI networks. These include combinational circuits and also sequential circuits which are partitioned by scan paths. Consequently, the circuit-under-test (CUT) will be considered combinational. Equally important, this analysis is confined to be linear in time complexity so it can be used efficiently to predict fault coverage. The following sections will start with an outline of the suggested procedure, to be followed by a detailed account of the various steps involved. Experimental results will follow.

2. THE PROCEDURE

The first issue that arises, is to seek the factors that influence the fault coverage in a BIST environment. For the purpose of this discussion, the environment will be that shown in Figure 1. The CUT is the circuit that needs to be tested, and has n inputs and

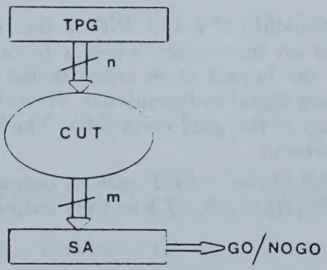


Figure 1. The BIST environment used in the procedure.

m outputs. The auxilliary hardware used for testing are the TPG (a pseudo-random test pattern generator), and the SA (a signature analyzer). When operating in the test mode, the TPG applies random vectors to the CUT and the responses are collected in the SA. The fault coverage depends on two factors. The first, is the effectiveness of the test set in sensitizing as many faults as possible to the primary outputs; the second is the error coverage of the signature analyzer resulting from the detected faults. These factors dictate the procedure that will be proposed. An analytical approach first determines the fault coverage obtained before compression of the output data. Then, an attempt is made to predict the error patterns that result from the faults in the CUT so that they can be used to estimate the fault coverage after compression.

The steps used in the procedure are the following:

(1) Extract a reduced equivalent fault list $\{f_1, f_2, \ldots, f_z\}$, of single stuck-at faults in the CUT.

(2) Determine the detection probabilities of each fault given the application of N random test vectors.

(3) Estimate the fault coverage, f_c, before compression.

(4) Given the above test set, obtain the resulting output error matrix and map it into a signature.

(5) Calculate the final fault coverage of the BIST scheme.

It should be mentioned that some parts in the procedure were iteratively determined, being guided by experimental results where formal proof was elusive. The types of experiments performed and the results obtained are presented in Section 6. The details of the procedure follow, pointing out in each case the assumptions and approximations involved.

3. DETERMINING THE DETECTION PROBABILITIES OF FAULTS

Since in the context of BIST a random stimulus is assumed, any of the methods in [2], [7] or [8] can be used to calculate the detection probabilities of each fault. In this procedure the COP testability program ([2]) was chosen for the following two reasons:

(1) Experiments by the authors of COP indicate there is good correlation with fault simulation.

(2) It is linear in time and space with the number of gates in a circuit.

COP computes the probability of detection $p_d(f_i)$ of a fault f_i on a line, as the product of the controllability and the observability values of that line. For example, the *detectability* (i.e. the probability of detection) of a line, l, stuck-at-1 is:

$$p_d(l|1) = C0(l)B(l) \tag{1}$$

where $C0(l)$ is the 0-*controllability* of l and $B(l)$ is the observability of l. The COP testability measures contain an inaccuracy when a fanout node reconverges since it assigns an observability to the fanout stem equal to the union of the observabilities of its branches, thus assuming signal independence. Nevertheless, this assumption only slightly penalizes the accuracy of the prediction ([4]). The following heuristic was added to COP to obtain more accuracy:

Heuristic: *Given a fault, f_i in a CUT with m outputs, with controllability $C(f_i)$ and observabilities $B_1(f_i), B_2(f_i), \ldots B_m(f_i)$ at each output respectively, the probability of detecting that fault is,*

$$p_d(f_i) = C(f_i)\left[(1-\alpha)max(B_1(f_i), B_2(f_i), \ldots B_m(f_i)) + \alpha \bigcup_{j=1}^{m} B_j(f_i)\right] \tag{2}$$

α is determined by the extent to which signals in the CUT are reconvergent. While it is unclear how to obtain α theoretically, experiments have indicated that if the circuit is decoupled, a value $\alpha = 1$ yields good correlation with fault simulation. On the contrary, if there is even a small degree of reconvergent fanout, a value closer to zero should be used. In other words, the value of α does not vary linearly with the degree of independence between signals in a circuit. As will be seen from experimental results, the heuristic above is critical in obtaining accurate predictions.

4. ESTIMATION OF FAULT COVERAGE BEFORE COMPRESSION

With the probability of detection of each fault computed, it is now desirable to estimate the resulting fault coverage before compressing the outputs into a signature.

Given a fault f_i, with probability of detection $p_d(f_i)$, the probability of detecting it with N random vectors is given by,

$$p_d(f_i)|_N = 1 - (1 - p_d(f_i))^N \tag{3}$$

In general, given N random vectors and z faults, assuming that the fault detection probability of each vector is statistically independent, the fault coverage can be predicted, and is given by,

$$\begin{aligned} f_c &= \frac{1}{z}\sum_{i=1}^{z}\left\{1 - (1 - p_d(f_i))^N\right\} \\ &= 1 - \frac{1}{z}\sum_{i=1}^{z}(1 - p_d(f_i))^N \end{aligned} \tag{4}$$

5. MAPPING THE ERRORS INTO A SIGNATURE

In this section, an output error polynomial will be defined and the degree of masking of this polynomial by the signature analyzer will be determined. This masking will reduce the fault coverage, f_c, given in Equation (4). For clarity, the analysis will begin with the special case of a single output CUT and will then be extended to the general m-output circuit.

For a single-output CUT with N vectors applied, the response R, may be written as a polynomial,

$$R(x) = r_0 x^{N-1} + r_1 x^{N-2} + \dots + r_{N-2}x + r_{N-1} \tag{5}$$

where r_i is the binary value of the output when vector i is applied. Similarly, the fault-free response R_{ff} to N vectors is,

$$R_{ff} = r_{0_{ff}} x^{N-1} - r_{1_{ff}} x^{N-2} + \dots + r_{N-2_{ff}} x + r_{N-1_{ff}} \tag{6}$$

Now, an error bit, e_i, may be defined as,

$$e_i = r_i \oplus r_{i_{ff}}$$

$$where\ e_i = \begin{cases} 0, & \text{if bit i is fault-free;} \\ 1, & \text{if bit i is faulty.} \end{cases}$$

Thus an error polynomial, E_s, results in the case of a single-output circuit with N vectors applied,

$$E_s(x) = e^{(0)} x^{N-1} + e^{(1)} x^{N-2} + \dots + e^{(N-2)} x + e^{(N-1)} \tag{7}$$

The above equation may be generalized for an m-output circuit, producing, for the i^{th} vector, the following polynomial:

$$E^{(i)}(x) = e_0^{(i)} + e_1^{(i)}x + e_2^{(i)}x^2 + \ldots + e_{m-2}^{(i)}x^{m-2} + e_{m-1}^{(i)}x^{m-1} \tag{8}$$

where $e_j^{(i)}$ is the error bit at output j when vector i is applied. Combining the above equations, one obtains the cumulative error matrix E_c,

$$E_c = \begin{pmatrix} E^{(0)} \\ E^{(1)} \\ \vdots \\ E^{(N-1)} \end{pmatrix} = \begin{pmatrix} e_0^{(0)} & e_1^{(0)} & e_2^{(0)} & \cdots & e_{m-1}^{(0)} \\ e_0^{(1)} & e_1^{(1)} & e_2^{(1)} & \cdots & e_{m-1}^{(1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_0^{(N-1)} & e_1^{(N-1)} & e_2^{(N-1)} & \cdots & e_{m-1}^{(N-1)} \end{pmatrix} \tag{9}$$

This error matrix may also be written in terms of a polynomial $E_c(x)$ representing the cumulative error after N vectors are applied. It should be kept in mind that with the application of each vector, the signature register is shifted.

$$\begin{aligned} E_c(x) = & (e_0^{(0)} + e_1^{(0)}x + e_2^{(0)}x^2 + \ldots + e_{m-1}^{(0)}x^{m-1})x^{N-1} + \\ & (e_0^{(1)} + e_1^{(1)}x + e_2^{(1)}x^2 + \ldots + e_{m-1}^{(1)}x^{m-1})x^{N-2} + \ldots \\ & (e_0^{(N-2)} + e_1^{(N-2)}x + e_2^{(N-2)}x^2 + \ldots + e_{m-1}^{(N-2)}x^{m-1})x + \\ & (e_0^{(N-1)} + e_1^{(N-1)}x + e_2^{(N-1)}x^2 + \ldots + e_{m-1}^{(N-1)}x^{m-1}) \end{aligned} \tag{10}$$

Now the values of each of the coefficients of x above should be determined so that when the polynomial $E_c(x)$ is compressed, the fault coverage can be estimated. The number of $1's$ (that is, fault derived errors) in the error matrix depends on two factors. First, given a number of random vectors, it depends on how many of these vectors detect each fault. This reflects on the number of *spoiled* rows in E_c (where a spoiled row is one in which at least one $e_j^{(i)}$ in that row has a value '1'). Second, if the fault is detected, the number of output bits in error must be ascertained since it affects the number of $1's$ in each spoiled row, and hence affects the compression.

It can be easily proved that the number of vectors that detect a fault, f_i, given N vectors, is $Np_d(f_i)$. The problem at hand is to determine **in general**, the number of vectors that detect **all** faults in the CUT. It is observed that in a reasonably complex circuit, the value of p_d varies significantly from fault to fault, and the standard deviation of the distribution of p_d is quite large. Further, owing to the random nature of the stimuli, it is not possible to be conclusive about the specific faults that are detected by a particular vector. With this in mind, an average measure, p_{dm}, is taken to represent the probability of detection of the faults in the circuit,

$$p_{dm} = \frac{1}{z}\sum_{i=1}^{z} p_d(f_i) \qquad (z = total\ number\ of\ faults) \tag{11}$$

Hence, with N vectors applied, the number of times, N_d, that each fault is detected **on the average**, is,

$$N_d = N \cdot p_{dm} \tag{12}$$

N_d can serve as a quick guide in estimating the number of times that the test set detects the faults. As will be explained in more detail in the following section, the correlation between N_d and the average number obtained by simulation is quite good.

To estimate the number of output bits in error would involve a detailed analysis of the circuit topology to determine the degree of independence of signals propagating from a fault to the primary outputs. For example, let a fault, f_i, have detection probabilities p_{d1}, p_{d2}, p_{d3}, at three outputs: if these probabilities are disjoint, it can be concluded that the fault is detected with probability $\sum_{j=1}^{3} p_{d_j}$ and **only one** output bit is in error. On the other hand, some vectors will propagate the faulty result to more than one output, and it is not possible to determine the degree of intersection between the above probabilities. Thus, when all faults are considered, in general, no reliable estimations can be made for a circuit. As a result, one has to rely on experimental data to estimate the number of erroneous outputs that a fault produces. Based on experiments on combinational circuits, the following conjecture is made:

Conjecture: *The average number of output bits in error produced by most single stuck-at faults in combinational circuits varies mostly between* 1 *and* 4. *Further, they do so unevenly, with the distribution shown in Figure* 2(*a*).

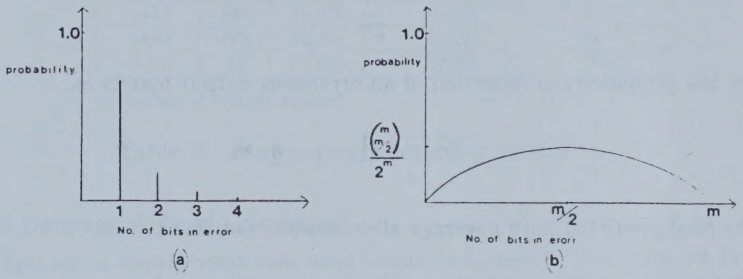


Figure 2. Distribution of the average number of erroneous output bits.
(a) Measured values, (b) Homogeneous case.

From this conjecture and Figure 2(*a*) it is observed that having only one output in error is by far the most likely occurrence. The curve then converges sharply to zero and indeed there is a very small probability of having more than four outputs in error when a fault is chosen at random. This can be contrasted with Figure 2(*b*) which depicts the homogeneous case in which all possible errors are equally likely. More details of the output errors produced by faults will be presented in the next section.

What remains is to use Equation (12) and the preceding conjecture to assign values to the error matrix, E_c, of Equation (9). Then, E_c will be compressed and a measure of the overall fault coverage can be predicted.

Again, estimating the actual distribution of the spoiled N_d rows is not straightforward. This is due to the randomness of the excitations and the different ways in which faults are manifested at the outputs. As a result, a *biased* probability model might only barely characterize the actual distribution. Also, keeping in mind that N_d is in itself a statistic, further complicates the task. Consequently, it will be assumed that the N_d spoiled output vectors can occur with equal likelihood at any time upon application of the N random input vectors. This leads to $\binom{N}{N_d}$ possible different combinations in which the spoiled output vectors may appear before they are compressed.

Similarly for the output bits in error, while empirically it has been shown that they are confined to a small number, it is unclear how the erroneous bits will be distributed at each primary output. Hence a further assumption must be made that given a spoiled output vector, there are $\binom{m}{m_s}$ ways in which they can appear, m_s being the number of spoiled bits. As determined experimentally, while m_s is bounded by 1 and m, it is most likely to be 1, with a progressively much smaller probability of being 2, 3 or 4.

With the above distribution in mind, the possible number of error polynomials can be expressed as:

$$\begin{aligned}\binom{N}{N_d}\binom{m}{m_s} \simeq \binom{N}{N_d}\binom{m}{1} \\ = \frac{N!}{N_d!(N-N_d)!}m\end{aligned} \tag{13}$$

Letting Equation (13) to be equal to λ, there are λ possible erroneous polynomials, as opposed to the homogeneous case where there are $2^{Nm} - 1$. **Note that** $(\lambda << 2^{Nm})$. It is further necessary to determine how the characteristic polynomial of the LFSR affects the mapping of the error polynomial, $E_c(x)$. Assuming that this mapping is uniform, and the LFSR is m bits wide $(\lambda > 2^m)$, the number of error polynomials producing the same signature as the fault-free case is,

$$\frac{\lambda}{2^m} \tag{14}$$

Therefore the probability of detection of an erroneous output matrix is,

$$1 - \frac{\lambda 2^{-m} - 1}{\lambda} = 1 - 2^{-m} \tag{15}$$

Hence the final predicted fault coverage after compression is,

$$f_c^{pred}(final) = f_c(1 - 2^{-m}) \tag{16}$$

Equation (16) is reminiscent to that obtained when all errors appearing at the output of the CUT are equally likely. It indicates that when the signature register length, m, is sufficiently large, the fault coverage before compression, f_c, can be taken as a reliable measure of the final fault coverage. Indeed, this is not a surprising result. What have been instructive are the intermediate results that were obtained, such as the number of output bits in error and the number of random vectors that detect a fault. The importance here is that the number of possible fault derived errors is limited to a very small subset of the total number of errors (which in fact never occur). This provides additional confidence in signature analysis. The next section will verify if the theoretical predictions presented above resemble the results obtained by fault simulation. This will determine if the initial objective, that of closely estimating fault coverage in BIST schemes in linear time, can be met.

6. EXPERIMENTS AND RESULTS

Three different classes of experiments were performed, each of which use fault simulation. The first set computes the fault coverage of various circuits without compression. The second looks at the resulting error patterns that are caused by the faults in the circuits. Finally, the third set measures the loss in fault coverage after the responses are compressed into a signature. All the experiments were executed on a VAX 11/750.

The circuits used were strictly combinational, though as has already been stated, the results can be generalized to sequential networks which incorporate scan design. Four circuits were used ([3]), varying in size from 192 to 1355 lines, the characterstics of which are illustrated in Table 1. While four seems like a small number, the circuits were reasonably diverse, varying from 'almost decoupled' (as in the $c880$), to 'very reconvergent' (as in the $c432$). Conclusive results were obtained in almost all cases.

circuit name	total gates	total lines	input lines	output lines	fanout stems	faults †	avg fanout	avg fanin
alu4*	63	192	14	8	31	237	3.71	2.43
c432	160	432	36	7	89	524	2.65	2.10
c880	383	880	60	26	125	942	3.50	1.90
c1355	546	1355	41	32	259	1574	2.97	1.95

† simplistically generated equivalent fault set (need not be minimal)

* circuit is the well-known 74181 4-bit ALU

Table 1. Characteristics of the combinational circuits.

Circuit	N†	f_c^{sim}	$f_c^{pred}(\alpha = 0)$	$f_c^{pred}(\alpha = 1)$
alu4	100	98.7%	99.1%	99.4%
c432	100	90.8%	94.2%	97.5%
c880	100	92.1%	89.7%	93.1%
c1355	60	81.0%	32.3%	77.0%

† Number of vectors applied

Table 2. Fault coverage without compression.

6.1 Fault Coverage Without Compression

The first set of experiments that were conducted were to determine the fault coverage of each circuit before compression. To be more precise, this is the fault coverage when random patterns are applied, since the idea is intended for BIST, where automatic test pattern generation should be avoided. The object here is to obtain the fault coverage using fault simulation and compare the results to those predicted by the analysis of the previous chapter.

The results of these experiments are shown in Table 2. To determine the number of vectors, N, that were applied, Equation (4) was used, trying different values of N until an adequate coverage was predicted. Then, these N vectors were actually simulated to obtain the fault coverage values shown in column 3, marked f_c^{sim}. With as few as 100 vectors, the fault coverage was very good, ranging from 90.8% to 98.7%. Even for the $c1355$, with N equal to 60, the fault coverage was still satisfactory at 81%.

The fault coverage values predicted by COP using Equation (4) are shown in columns 4 and 5. There are two values for the predicted fault coverage. This is in order to verify if the heuristic of Section 3. Column 4 shows the predicted fault coverage when the probability of detecting a fault is taken as the maximum detection probability at the output of the CUT. In the heuristic, this corresponds to a value of $\alpha = 0$. Column 5 is the estimated coverage when signal independence is assumed, and the detection probability of a fault is the joint probability of detection at all the outputs ($\alpha = 1$). Indeed, the heuristic is justified for three of the four circuits. In the case of the $alu4$ and the $c432$, both of which contain considerable reconvergent fanout, the fault coverage values of column 4, with α equal to zero, approximate the actual values of f_c very well (the percentage error is 0.4% for the $alu4$ and 3.6% for the $c432$). For the $c880$, a very decoupled circuit, the value in column 5 ($\alpha = 1$), is almost exactly equal to that obtained by simulation (1.08% discrepancy). In the case of the $c1355$, it seems that COP predictions fail due the small number of vectors applied.

It may be concluded from the results shown in Table 2, that Equation (4), coupled with the appropriate value for α, yields close correlation with results obtained by fault simulation.

6.2 Error Characteristics

The experiments in this section were conducted so as to obtain two measures. The first is to ascertain the number of times that a fault was detected given a random test set; the second is to find the number of output bits that are in error due to a fault. These two factors are dominant in forming the output error matrix, which in turn, influences the subsequent masking of the errors by the signature analyzer.

To obtain a measure of the correlation between the *predicted* number of times, N_d^{pred}, that a fault is detected by the vector set, and the experimental value, N_d^{fsim}, as measured by fault simulation, a correlation coefficient, cc, is defined below. This coefficient ranges between 0 and 1 where a value closer to 1 indicates that theory correlates well with practice.

Let the variable, x, represent the number of vectors that detected a fault, f_i, by fault simulation. With z faults in the fault list, there are z values of x, $\{x_1, x_2, \ldots, x_z\}$.

Similarly, let the variable y represent the number of vectors that detect a fault as predicted by $N_d^{pred}(f_i) = Np_d(f_i)$.

The normalized covariance, $\frac{cov(x,y)}{\sigma_x \sigma_y}$, can be taken as the correlation coefficient, cc, where σ_x and σ_y are the standard deviations of x and y respectively. It should be emphasized that the values of x were averaged over a number of independent experiments.

The correlation coefficients for the various circuits are shown in Table 3. Column 2 has the values $cc^{(1)}$ where for y, the output with the *maximum* probability of detection was taken ($\alpha = 0$). These coefficients indicate that good correlation exists between the values predicted and those actually obtained. Indeed, for most faults, the discrepancy was less than one vector. In column 3, $cc^{(2)}$ lists the degree of correlation when the joint probabilities of detection are taken, corresponding to $\alpha = 1$. It is observed that for the $c880$, $cc^{(2)}$ is higher than $cc^{(1)}$, again reflecting the fact that a value of α close to one yields better predictions for circuits without many reconvergent signals. The $c1355$ has a higher degree of reconvergence than the $c880$, and while $cc^{(2)}$ indicates a correlation factor of 0.74, a value of α closer to zero gives a more accurate prediction. Finally, the first two circuits, $alu4$ and $c432$, have enough reconvergent gates to render the coefficient $cc^{(2)}$ meaningless, and thus a value is not given.

Circuit	$cc^{(1)}(\alpha = 0)$	$cc^{(2)}(\alpha = 1)$
alu4	0.87	-
c432	0.66	-
c880	0.69	0.72
c1355	0.76	0.74

Table 3. Correlation coefficient on a fault-by-fault basis.

Circuit	1	2	3	4	> 4
alu4	64%	22%	8%	4%	2%
c432	51%	19%	15%	10%	4%
c880	88%	9%	1.5%	1%	0.5%
c1355	94%	5%	0.75%	0.25%	0.0%

Table 4. Distribution of output bits in error.

The number of faulty output bits measured when a fault is detected are shown in Table 4. It can be observed that the most likely occurence when a single stuck-at fault is detected is having one output bit in error. Further, the probability of having more than four erroneous outputs is almost negligible. It is worth noting here that even for the larger circuits which have more than 25 outputs each, a fault spoils only a small number of bits. As was stated in Section 5, it is difficult to predict with confidence the number of output bits that are in error. Consequently, in this case, the experimental data cannot be compared to any theoretical predictions.

To conclude, the results obtained in this section give a good indication of what error patterns to expect when single faults are considered. It is interesting to observe that on the circuits used, *on the average*, between 8% and 25% of the random vectors applied detected each fault, and when this occurred only 1 or 2 output bits were spoiled in most cases.

6.3 Fault Coverage With Compression

The final sets of experiments performed compute the fault coverage of a CUT after compression. The responses to the random stimuli are now compressed in a Multiple-Input Signature Register (MISR). The MISR has length m, equal to the number of primary outputs of the circuit. For each circuit, three different characteristic polynomials for the MISR are used, the final fault coverage being the average of the three. These characteristic polynomials are all irreducible and primitive. The results are shown in Table 5. Column 3 lists f_c^{fsim} $(final)$, the final fault coverage after compression, obtained experimentally. Column 4 lists the value of fault coverage, f_c^{pred} $(final)$, predicted by Equation (16) ($f_c^{pred}(final) = f_c(1 - 2^{-m})$). Note that f_c on the right side of this equation is the fault coverage predicted *before compression*. Column 6 shows the percentage of masked faults. These are the faults that were detected with fault simulation before compression, but were masked by the MISR.

Circuit	N†	$f_c^{fsim}(final)$	$f_c^{pred}(final)$	error	Faults masked
alu4	100	98.4%	98.3%	0.1%	0.33%
c432	100	89.7%	93.8%	4.5%	0.95%
c880	100	92.1%	93.1%	1.1%	0.0%
c1355	60	81.0%	77.0%	4.9%	0.0%

† Number of vectors applied

Table 5. Fault coverage after compression.

Table 5 indicates that the error between the estimated and measured values is less than 5% in all cases. This discrepancy is almost entirely due to the value of f_c obtained by Equation (4) and not because of the masking in the signature register.

7 CONCLUSIONS

The primary objective of the work in this paper was to determine whether a measure of the fault coverage in BIST schemes could be estimated confidently without resorting to fault simulation, which is very costly.

While a simple *YES/NO* answer has not clearly emerged, the results obtained indicate that the objective, to a large extent, was met. Indeed, following the suggested procedure, the predictions approximated the experimental results very closely. This can be attributed to two reasons. First, estimating the random-pattern fault coverage of a circuit without compression is quite accurate when using the COP testability measures together with the new heuristic which accounts for the observabilities of faults at primary outputs. The second reason is merely due to the strength of signature analysis. If the characteristic polynomial is carefully chosen, the probability of masking fault-derived errors is very small, and Equation (16) agrees very well with experimental data.

From the results obtained it can be surmised that the fault coverage in BIST depends more on the effectiveness of the random vectors in sensitizing faults to the outputs of the CUT, than it does on the signature analyzer's ability to compress data without losing too much information. This is clearly evident in the fact that while a sufficiently large random test set does not detect as much as 5-15% of all irredundant faults, the signature register accounts for only 1-2% reduction in fault coverage. Further, it is up to the designer to choose a MISR which is long enough and which has an irreducible and primitive characterstic polynomial so as to minimize masking. With regard to the loss in fault coverage due to random stimuli, if the CUT cannot be modified to be random-pattern testable, a large part of the loss in fault coverage can be recovered by providing for deterministic vectors on the *hard* faults (these faults are indicated by the testability measures). The deterministic patterns will only be a small subset of the those applied, and can therefore be stored in a ROM on-chip without penalizing the area overhead significantly. Now, with the availability of deterministic vectors, the major cause of the loss in fault coverage can be attributed to masking by the compressor. To rectify this loss, Zorian ([9]) proposes to modify the output data. The functional information of the CUT is exploited to select a compressor function that has a *much* higher detection probability than conventional MISRs.

The main advantage of the suggested procedure is that it can attain sufficient accuracy in estimating the fault coverage in a time that is linear. It has also been instructive in providing some statistics which are useful for random testing in general. These include the number of output lines that are in error when a fault is sensitized to the circuit outputs, and also the percentage of random vectors that detect each fault on the average.

REFERENCES

[1] V.D. Agrawal, S.C. Seth, and C.C. Chuang, "Probabilistically Guided Test Generation", *Proc. Int. Symp. on Circuits and Systems*, June, 1985.

[2] F. Brglez, "On Testability Analysis of Combinational Networks", *Proc. of IEEE Int. Symp. on Circuits and Systems*, May 1984, pp. 221-225.

[3] F. Brglez and H. Fujiwara, "A Neutral Netlist of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN ", *Proc. 1985 Int. Symp. on Circuits and Systems*, June 1985.

[4] F. Brglez, P. Pownall, and R. Hum, "Applications of Testability Analysis: From ATPG to Critical Delay Path Tracing", *Proc. 1984 Int. Test Conference*, Oct. 1984, pp. 705-712.

[5] N. Cotsapas, "Analysis of Fault Coverage Masking in Built-In Self-Test Schemes", *M.Eng. Thesis, McGill University*, July 1985.

[6] J.E. Smith, "Measures of the Effectiveness of Fault Signature Analysis", *IEEE Trans. Comp., vol C-29*, June 1980, pp. 510-514.

[7] J. Savir, G.S. Ditlow and P.H. Bardell, "Random Pattern Testability", *IEEE Trans. Comp. vol C-33*, Jan. 1984, pp 79-90.

[8] S.S. Seth, L. Pan, and V.D. Agrawal, "PREDICT: Probabilistic Estimation of Digital Circuit Testability", *Proc.* 15^{th} *Int. Symp. on Fault Tolerant Computing*, June 1985, pp. 220-225.

[9] Y. Zorian and V.K. Agarwal, "Higher Certainty of Error Coverage by Output Data Modification", *Proc. 1984 Int. Test Conf.*, October 1984, pp. 140-147.

MICROPROCESSOR TESTING

ZHIJUN TONG, LARRY L. KINNEY, RICHARD Y. KAIN
Electrical Engineering Department, University of Minnesota
Minneapolis, MN 55455

ABSTRACT

In this paper a method is presented to generate tests to find faults in the instruction execution process and register decoding functions within microprocessors. For checking the instruction execution process, the approach is to load all registers with codewords such that the fault will cause erroneous results. First the functions of the READ register and WRITE register instructions are tested, then it is verified that all registers can be loaded with codewords, and lastly the remaining instructions are tested. It is shown that register decoding faults can also be detected with the use of codewords.

I. INTRODUCTION

With the development of integrated circuit technology, more circuits than ever can be built on a small chip of silicon. In a VLSI device, such as a microprocessor chip, there are thousands of gates and storage elements on a single chip. The detailed circuitry in a VLSI chip is usually unknown to the users. Hence, conventional test generation methods which make use of detailed circuit information are not feasible. This situation has caused people to devote significant attention to functional testing. Over the past few years a variety of techniques have been proposed to test VLSI devices [2], [3], [4], [5].

Recently, Brahme and Abraham [1] presented a new fault model for both instruction execution sequencing and control section logic. In their testing strategy one first tests the register READ instructions, and then the remaining instructions are tested in sequences using the READ register instruction. This strategy does not generate an effective test for the linked faults arising from data manipulations between registers. In [1] the register READ instruction must include saving and recovering the contents of registers which are on the path from the tested register to the OUT node, where data is externally observable.

In Section III, we develop a strategy to generate codewords useful in detecting complex faults. We also develop a new strategy to derive tests for faults in the instruction execution process based on the fault model developed in [1]. Our approach has higher fault coverage and the test generation procedure is simpler than in [1]. We also show how to use codewords to detect register decoding faults. Register decoding faults can be detected using the same procedure used to test the instruction execution process.

II. THE FUNCTIONAL FAULT MODEL

To facilitate the generation of tests for faults in microprocessors, we need to develop functional fault models. We also decompose the microprocessor into a set of functional units such as registers, register decoding logic, data transfer path, ALU, and instruction sequencing logic, as in [1]. We present the fault model for register decoding and instruction sequencing.

Due to a register decoding fault, when register R_i (of the n_R processor registers) is to be selected, any of the following may actually happen: 1) No register is selected: $f_D(R_i) = \phi$; 2) Register R_j is selected instead of R_i; $f_D(R_i) = R_j$, where $R_i \neq R_j$; 3) More than one register is selected possibly including R_i: $f_D(R_i) = \overline{R}$, where $\overline{R}$ is a subset of registers in the microprocessor, and $|\overline{R}|>1$. The notation is further extended for multiple registers:

$$f_D(R_1,R_2,\ldots,R_n) = \cup_{i=1}^{n} f_D(R_i).$$

Due to an instruction execution sequence fault, any of the following may occur:
1) One or more microoperations which are normally active become inactive; 2) Microoperations which are normally inactive become active; 3) A set of microinstructions which are normally inactive becomes active; 4) A set of microinstructions which are normally active becomes inactive.

The following conditions are assumed for this study:
1) No faulty microoperation which accesses memory or any device external to the microprocessor occurs; 2) Instruction faults do not create an infinite loop; 3) An external tester monitors all microprocessor chip pins. If any fault addresses memory or external devices improperly, it can be detected by observing the information on the pins. Also an infinite loop can be detected since the faulty program will run past a known time limit.

III. TEST GENERATION

In this section we present test generation procedures for register decoding faults and instruction execution process faults. First we require some terminology and then we prove some basic lemmas concerning fault effects.

Definition 1: $\text{WRITE}(R_i) = \{I_{1,i},I_{2,i},\ldots,I_{n_i,i}\}$ is the shortest sequence (of instructions $I_{j,i}$) that copies data from sources (memory or devices external to the microprocessor) into register R_i.

Definition 2: $\text{READ}\ (R_i) = \{\overline{I}_{1,i},\overline{I}_{2,i},\ldots,\overline{I}_{n_i,i}\}$ is the shortest sequence (of instructions $I_{j,i}$) that copies data from register R_i to a destination external to the microprocessor.

Definition 3: $R_W = \{R_{1,W},R_{2,W}\ldots,R_{n_R,W})$ is a permutation of the register set such that R_i lies on the left of R_j if and only if performing WRITE (R_j) will not affect the contents of R_i if the processor is fault-free. A similar ordering rule applies to $R_R = \{R_{1,R},R_{2,R},\ldots,R_{n_R,R}\}$ for READ.

For convenience, we use WRITE(R) to denote instruction sequences $\text{WRITE}(R_{i,W})$ for $i = 1,2,\ldots,n_R$, and READ(R) to denote instruction sequences $\text{READ}(R_{i,R})$, for $i = 1,2,\ldots,n_R$.

Definition 4: $S(I_i) = \{R_{1,i},\ R_{2,i},\ \ldots,\ R_{p_i,i}\}$ denotes the set of source registerse which provide the operands for instruction I_i.

Definition 5: $D(I_i) = \{\overline{R}_{1,i},\ \overline{R}_{2,i},\ \ldots,\ \overline{R}_{p_i,i}\}$ denotes the set of destination registers which are changed by executing I_i.

Similar definitions can be developed for composite instructions, microinstructions, and microoperations:

1) $D\{I_1,\ I_2,\ldots,I_n\} = \cup_{i=1}^{n} D(I_i)$,
2) $S\{I_1,\ I_2,\ldots,I_n\} = \cup_{i=1}^{n} S(I_i)$,

where $\{I_1,\ I_2,\ldots,I_n\}$ denotes the instruction sequence for a composite instruction.

Lemma 1: Let the composite instruction $\text{WRITE}(R_i) = \{I_{1,i},I_{2,i},\ldots,I_{n_i,i}\}$. After the execution of $\text{WRITE}(R_i)$ with data d, all registers in the set $f_D(D(\text{WRITE}(R_i)))=f_D(D(I_{1,i},I_{2,i},\ldots,I_{n_i,i}))$ will be loaded with data d if $f_D(D(I_{ji})) \neq \phi$ for all I_{ji}; in the WRITE(R_i).

Lemma 2: Let READ(R_i) = $\{\bar I_{1,i}, \bar I_{2,i}, \ldots, \bar I_{n_i,i}\}$ and assume the composite instruction READ(R_i) is executed. Then

1) If $f_D(S(\bar I_{k,i})) \neq \phi$ for all $\bar I_{k,i} \varepsilon$ READ(R_i), then $\bigcup_{k=1}^{n} R_k$ or $\bigcap_{k=1}^{n} R_k$ will be read out, depending on technology, where $\{R_1, R_2, \ldots, R_n\}$ denotes the set $f_D(R_i)$.

2) If $f_D(S(I_{i,k})) = \phi$ for some $I_{i,k} \varepsilon$READ(R_i), then the all zero or all one vector will be read out, depending on technology.

Proof: The proofs of Lemma 1 and Lemma 2 are given in [2].

Lemma 3: Let $D = \{d_1, d_2, \ldots, d_n\}$ be the set of binary numbers such that for any two numbers $d_i, d_j \varepsilon D, i \neq j$, $d_i \$ d_j = d \notin D$, where \$ represents the OR or AND operation. Then, for any subset $\bar D = \{\bar d_1, \bar d_2, \ldots, \bar d_{\bar n}\}$ of D,

$\bigcup_{i=1}^{\bar n} \bar d_i = \bar d \notin D$ and $\bigcap_{i=1}^{\bar n} \bar d_i = d \notin D$ for $|D| > 1$.

Proof: Straightforward.

Definition 6: A fault is a single fault if at most one additional microoperation is executed as a consequence of the fault. A single fault instruction can be expressed as

$F(I_i) = I_i - M^- + O^+$

where M^- denotes the set of missing microinstructions and O^+ denotes the additional microoperation.

Definition 7: A fault is a single repeated fault if at most one additional microoperation is enabled, but is executed more than once. A single repeated fault can be expressed as

$F(I_i) = I_i - M^- + O^*$

We use the MC 68000 microprocessor as an example (as in [1]). The basic microorder set of MC 68000 is shown in Table I. This is also the set of single faults.

TABLE I

Set of microorders for MC 68000

SWAP, CLR, NOT, NEG, BTEST, BSET, BCLR, BCHG, SHIFTLEFT, SHIFTRIGHT, MOV, AND, OR, EOR, EXG, ADD, MUL, SUB, DIV.

Definition 8: An n-linked fault is a set of single faults $M^+ = \{O_1, O_2, \ldots, O_m\}$ involving n different registers. The n registers must include all operand sources and result destinations for all the single fault operations. A linked fault can be represented by a graph whose n nodes correspond to the n registers and which has a set of branches, with a branch for each fault operation extending from each operand source to each result destination. The graph for a single linked fault must not contain any cycles, except for self-loops, and must be a single chain (plus self-loops) encompassing all n registers. This chain imposes a precedence ordering on the registers involved in the fault. If the graph of a single linked fault does not contain any self-loops, then the fault is a simple linked fault.

Definition 9: Two single faults, O_i and O_j, are independent if and only if

$S(O_i) \cap D(O_j) = \phi$ and $D(O_i) \cap S(O_j) = \phi$.

Two linked faults,

$M^+ = \{O_1, O_2, \ldots, O_n\}$ and $\bar M^+ = \{\bar O_1, \bar O_2, \ldots, \bar O_n\}$.

are independent if and only if there does not exist a pair of single faults $O_i \varepsilon M^+$ and $O_j \varepsilon \bar M^+$ which are dependent.

Definition 10: A crossed fault is two linked faults,

$M^+ = \{O_1, O_1, \ldots, O_n\}$

and $\bar M^+ = \{\bar O_1, \bar O_2, \ldots, \bar O_n\}$,

such that there is only one register that satisfies the condition:

$R_i \ \varepsilon\ D(M^+) \cap S(\bar M^+)$ or $R_i \ \varepsilon\ S(M^+) \cap D(\bar M^+)$

In order to detect a fault, we have to find some instruction sequences and corresponding input patterns such that any faulty operation will show up in observations of the data at the microprocessor's pins. Many of our test sequences begin by loading all registers with selected data called codewords, which have the property that any specified faulty operation on those codewords should produce a non-codeword result.

However, if a microprocessor has many registers and a small word size, the codeword generation process becomes time-consuming, maybe even impossible. We present a new approach for codeword generation. The microoperations of a microprocessor are divided into two subsets: Set 1 contains those microoperations (Such as MOVE, EXCHANGE and AND) for which the result equals the operand(s) if all operands are equal. Set 2 contains the microoperations not in set 1. Thus, a trivial codeword set consisting of only one codeword can detect the faults in set 2.

Example: TABLE II is a codeword set for 6 registers of width 4 for testing the operations in set 1. TABLE III is a codeword set for 6 registers of width 4 for testing set 2 operations.

TABLE II

register #	codeword
1	0011
2	0101
3	1001
4	0110
5	1010
6	1100

TABLE III

register #	codeword
1	0010
2	0010
3	0010
4	0010
5	0010
6	0010

Our approach simplifies both the generation of codewords to detect complex faults and the analysis of the fault coverage of each codeword set. By splitting the instruction set and using a separate codeword set for each instruction subset, we can test the 6 4-bit register processor. On the other hand, if one codeword set had to be used, the processor could not be tested completely.

Lemma 4: Let instruction I_i have a simple n-link fault where each fault is allowed to occur once, ending at register R_k, and R_k has initial value zero. If I_i is executed n-1 times, and none of these executions changes the content of register R_k, then further execution of this instruction will not change the contents of R_k either.

Proof: The proof is by induction on n.

The basis of induction is the 2-linked fault. Let registers R_1 and R_2 be registers involving the linked fault as shown in Figure 1. The fault can be expressed as $R_2 \leftarrow R_1 \$ R_2$, where \$ represents one of the possible extra two-operand operations, which are MOV, ADD, OR, EOR, SUB, MUL, AND, or DIV. If the fault is AND, MUL, or DIV, it is clear that the content of R_2 will not change since R_2 has initial value zero. Consider the remaining possible faults. It is easily seen that R_2 remains unchanged if and only if R_1 contains zero. Hence, after execution of I_i if R_2 is not changed, then either R_1 contains zero or the fault is AND, MUL, or DIV. So further execution of I_i will leave the contents of R_2 unchanged.

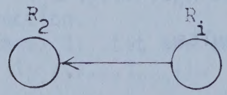


Fig. 1. A 2-linked fault

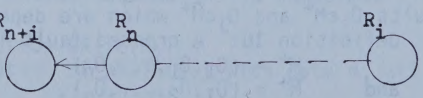


Fig. 2. An n+1-linked fault

Assume that Lemma 4 is true for a simple n-linked fault. Consider an n+1-linked fault. Let $R = \{R_1, R_2, \ldots, R_{n+1}\}$ be the set of registers involved in the n+1-linked fault as shown in Fig. 2. If the last stage of the fault, $R_{n+1} \leftarrow R_n \$ R_{n+1}$, is AND, MUL, or DIV, the Lemma is obviously true. If the fault is not AND, MUL, or DIV, and I_i is executed n times and it is observed that the content of R_{n+1} is not changed, then R_n must have been zero between the first and last execution of the faulty microoperation, $R_{n+1} \leftarrow R_n \$ R_{n+1}$, as we argued in the proof of the basis. This observation also shows that if I_i were executed n-1 times, none of those executions would change the content of register R_n, providing it had started with an initial value of zero. From the hypothesis of the induction we know that further executions of I_i will leave register R_n unchanged. This also keeps R_{n+1} unchanged. So the Lemma is proved.

The definitions and lemmas developed in this section are very important for test generation. Now we will present test procedures for register decoding faults and instruction execution faults.

Our first test procedure verifies that the READ and WRITE register instructions perform their intended functions. In all procedures, every READ operation includes a check that the result is consistent with correct operation.

```
                            Procedure 1
                       For i=1 to $n_R$ do
step 1:                {WRITE($R_i$) with zero; READ($R_i$) $n_R$ times}
step 2:                  WRITE($R_i$) with one; READ($R_i$)
step 3:                {WRITE($R_i$) with zero; READ($R_i$) $n_R$ times}
                       END {for}
```

Theorem 1: Suppose that the sequences WRITE(R_i) and READ(R_i) can have at most a simple linked fault to R_i (but any kinds of fault to other registers), and that no faults are detected in procedure 1. Then all registers R_i can be loaded with desired data and read out by executing WRITE(R_i) and READ(R_i).

Proof: First consider the case where register R_i can not be loaded by executing WRITE(R_i). Furthermore, assume that in executing WRITE(R_i) and READ(R_i) there is no fault which will change R_i. Let R_i have initial data d_0. When d_0 is not zero, the fault can be detected while executing READ(R_i) in step 1. If d_0 is zero, then the fault will be detected in step 2 since the desired data becomes one in step 2.

Next consider the case in which register R_i cannot be loaded by executing WRITE(R_i), but there is a simple linked fault to register R_i in executing WRITE (R_i) or READ(R_i). Since the WRITE(R_i) and READ(R_i) are executed alternately n_R times in step 1, then if no fault is detected in step 1, the linked fault will keep the data zero in R_i (Lemma 4) and the fault is detected in step 2.

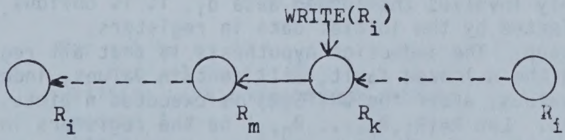


Fig. 3. A linked fault in the execution of WRITE(R_i)

Consistent with the hypothesis, there is one other case in which register R_i cannot be loaded by executing WRITE(R_i); there is a simple linked fault which leads R_i, and some register R_k which is in the linked fault is loaded by executing WRITE(R_i) (see Fig. 3). Assume that the operations between R_i and R_k in the linked fault graph are not AND, MUL or DIV. After the execution of the instruction sequence in step 1, the registers between R_i and R_k (in the linked fault order) are zero if no faults are detected in step 1. Then by induction it is easy to verify that by executing the instruction sequence in step 2, the desired data will be read out if and only if all registers between R_i and R_k (excluding R_k) were one. Since R_i cannot be loaded by WRITE(R_i), some operations between R_i and R_k must not be MOVE. Without loss of generality, assume the operation, $R_m \leftarrow R_m \$ R_k$, is not MOVE. Then the fault can be detected in step 3 since R_m has one and R_k has zero in step 3 and R_m cannot be changed to zero. So the Theorem is proved.

```
                          Procedure 2
         For i=1 to $n_R$ do
step 1   WRITE(R) with codewords $n_R$ times;
step 2   READ($R_i$)
               End {for}
```

Theorem 2: Suppose that the WRITE(R_i) and READ(R_i) for i=1,2,...,n_R can perform their intended functions, that WRITE(R) may have only independent single, repeated, linked and crossed faults, and that no faults are detected in executing Procedure 2. Then by executing the instruction sequence in WRITE(R) n_R times, all registers R_i will be loaded with the desired data d_i, independent of the initial values in of the registers.

Proof: We first prove that executing WRITE(R) n_R times guarantees that the final values in the registers will not depend on the initial values in the registers. We will detail the proof for linked faults only since single and repeated faults are special cases of linked faults, and crossed faults are couplings of two linked faults. The proof for linked faults is by induction over the degree of linking in the fault.

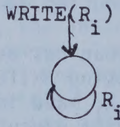


Fig. 4. A 1-linked fault

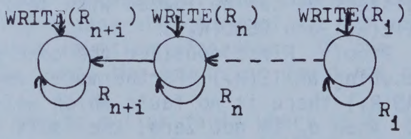


Fig. 5. A n+1-linked fault

Basis of induction: Assume only one register R_i is involved in the linked fault, as shown in Fig. 4. Consider the two cases after the register R_i is loaded with the desired data by executing WRITE(R_i). 1): No fault occurs. The register R_i will contain the desired data d_i. 2): Some fault microoperation, $R_i \leftarrow \$ R_i$, also does occur. Since the data manipulation only involves the loaded data d_i, it is obvious the result will not be affected by the initial data in registers.

Induction step: The induction hypothesis is that all registers, which are involved in the n-linked fault, will contain values, independent of their initial values, after the WRITE(R) is executed n times. Consider an n+1-linked fault. Let $R=\{R_1,R_2,...,R_{n+1}\}$ be the registers in the linked fault as shown in Fig. 5, where the edges that have no source denote the execution of the instruction WRITE(R_i).

The fault operation loading R_{n+1} can be shown as

$$R_{n+1} \leftarrow R_n \ \$ \ R_{n+1}$$

or

$$R_{n+1} \leftarrow \ \$ \ R_{n+1}$$

By the induction hypothesis the register set $\{R_1,R_2,...,R_n\}$ will contain values independent of their initial values after WRITE(R) is executed n times. Further execution of WRITE(R) will not produce any values in any R_i ($i=1,2...n$) which depend on the initial register values. So after R_{n+1} is loaded by $n+1$ executions of WRITE(R_{n+1}), the value in R_{n+1} will not depend on the initial register contents. In the worst case the n_R registers will be in the linked fault, but then executing WRITE(R) n_R times will load all registers with values independent of the initial register contents.

For a crossed fault, a proof similar to the one above for linked faults can be constructed.

Thus, it is proved that all registers will have data independent of initial value of registers at step 1 of procedure 2. We allow any number of single, repeated linked, or crossed faults if they are independent. If some register R_i is not loaded with the desired codeword d_i, it will be detected at step 2. Otherwise all registers will be loaded with the desired codewords. Hence, Theorem 2 is proved.

Now we present test procedures to detect faults in the instruction execution process.

```
    Procedure 3:  Testing for faults in READ register instructions
                   For i=1 to $n_R$ do
                       For j=1 to $n_R$ do
step 1:            WRITE(R) with codewords $n_R$ times;
step 2:            READ($R_i$);
step 3:            READ($R_j$);
                       End{j}
                   End {i}
```

```
    Procedure 4:  Testing for faults in WRITE register instructions
                   For i=1 to $n_R$ do
step 1:            WRITE(R) with codewords $n_R$ times;
step 2:            WRITE($R_i$);
step 3:            READ(R);
                   End {i}
```

```
    Procedure 5:  Testing for faults in all instructions $I_i$
                   For i=1 to $n_I$ do
step 1:            WRITE(R) with codewords;
step 2:            Execute $I_i$;
step 3:            READ(R);
                   End {i};
```

If the number of registers is large compared with the number of instructions, we can substitute the following Procedure for Procedures 3, 4, and 5.

Procedure 6

```
                   For i=1 to $n_I$ do
                       For j=1 to $n_R$ do
step 1:            WRITE(R) with codewords $n_R$ times;
step 2:            Execute $I_i$;
step 3:            READ($R_j$);
                       End {j};
                   End {i};
```

Procedure 6 combines Procedures 3, 4, and 5 into one. But the time complexity of Procedure 6 is $o(n_I x n_R^3)$ compared with $o(n_R^4+n_I x n_R)$ for Procedures 3, 4, and 5 together. Hence when $n_I >> n_R$, executing Procedures 3, 4, and 5 is preferred.

Theorem 3: If a set of codewords selected to detect logical fault operations AND or OR are loaded in Procedure 3, then all register decoding faults can be detected in Procedure 3.

Proof: The register decoding faults may be divided into several cases.

Case 1): $f_D(R_i)=\phi$ for some R_i. The fault will be detected by executing READ(R_i) in step 2 since all ones or zeros will be read out, depending on the technology, following Lemma 2.

Case 2): $f_D(R_i) = \overline{R} = \{\overline{R}_1, \overline{R}_2, \dots, \overline{R}_n\}$ for some R_i and $|\overline{R}| > 1$. According to Lemma 1 all registers in the set $f_D(D(\text{WRITE}(R)))$ will be loaded with data in the set of codewords, but it is not assured that register R_i has the desired data d_i. Let $D=\{d_1, D_2, \dots, d_{n_R}\}$ be the set of codewords selected to detect logical operation AND and OR faults. Let $\overline{d}=\{\overline{d}_1, \overline{d}_2, \dots, \overline{d}_n\}$ be the data set in the set $f_D(D(\text{WRITE}(R)))$. Obviously $\overline{D}$ is a subset of D. From the properties of codewords we know D satisfies the condition in Lemma 3 ($d_i \$ d_j \notin D$). If $|\overline{D}| = n_R$, then $D = \overline{D}$. The fault will be detected by executing READ(R) since data $d = \bigcup_{i=1}^{n} R_i$ or $d = \bigcap_{i=1}^{n} R_i$ will be read out, depending on technology, (following Lemma 2) and $d \notin D$ (following Lemma 3). If $|\overline{D}|<n_R$, then there is at least one data d_k such that $d_k \in D$ and $d_k \notin \overline{D}$. The fault will also be detected by executing READ(R_k) since d_k cannot be produced by an OR or AND fault on the set $\overline{D}$ (Lemma 3).

Case 3): $f_D(R_i)=R_j$ for all registers R_i. If there are two registers, R_i and R_j, such that $f_D(R_i) \cap f_D(R_j) = R_k$, then the fault will be detected by executing READ(R_i) or READ(R_j), depending on the content of register R_k, at the end of step 2 assuming R_i and R_j are associated with different codewords. If $f_D(R_t) \cap f_D(R_j) = \phi$ for $i \neq j$ and $i=1,2,\dots,n_R$, $j=1,2,\dots,n_R$, then the fault is undetectable since the fault is equivalent to reassigning addresses to the registers in the program model. So we have proved that Procedure 2 detects all detectable register decoding faults.

In this section we presented procedures for detecting faults in the register decoding function and in the instruction execution process. If we divided the set of microoperations into two subsets to generate the codeword set(s), procedures 2, 3, 4, and 5 should be executed once with each (different) codeword set.

IV. CONCLUSION

In this paper we presented test procedures to detect faults in the register decoding function and the instruction execution process of a computer. Since the fault model is general, independent of implementation detail, and the test generation procedures are very simple, the test generation procedure can be readily automated. We use the same strategy to detect faults in the instruction execution process as in [1], but our approach is more effective. Our coverage of linked faults is more general: in [1] they could only derive tests for those linked faults in which transfers between registers are made without data manipulation. The time complexity of the test sequence remains $o(n_R * n_I + n_R^4)$. The test generation procedure used to detect instruction execution process faults also detects register decoding faults, so the complete test sequences for the whole microprocessor system can be reduced by using these strategies. Furthermore, in our approach codeword generation is more flexible since the set of microoperations can be divided into subsets for codeword generation, which reduces the constraints on the codewords and therefore simplifies both their generation and the analysis of their coverage.

REFERENCES

[1] D. Brahme and I. A. Abraham, "Functional testing of microprocessors", IEEE Trans. Computer, vol. C-33, pp. 475-485, No. 6, June 1984.

[2] S. M. Thatte and J. A. Abraham, "Test generation for microprocessors", IEEE Trans. Computer, vo. C-29, pp. 429-441, May 1980.

[3] T. Sridhar and J. P. Hayes, "A functional approach to testing bit-sliced microprocessors", IEEE Trans. Computer, vol. C-30, pp. 563-571, Aug. 1981.

[4] C. Robach and G. Saucier, "Diversified test methods for local control units", IEEE Trans. Computer, vol. C-24, pp. 562-567, May 1975.

[5] C. Robach and G. Saucier, "Microprocessor functional testing", Digest of papers, 1980 Test Conference, pp. 433-443, 1980.

[6] Y. Min and Y. H. Su, "Test functional faults in VLSI", 19th Design Automation Conference, 1982.

A TEST PATTERN GENERATOR WITH EMBEDDED COMPRESSED OUTPUT FOR BUILT-IN SELF-TEST.

P.SHEEN* RAMASWAMI DANDAPANI** KEWAL K. SALUJA*
*Department of Electrical and Computer Engineering,University of Newcastle, NSW., 2308,Australia **Department of Mathematics and Computer Sciences,Youngstown State University,Youngstown,Ohio

ABSTRACT

This paper outlines a design procedure for test pattern generation and test output evaluation in Built-in Self-test. The chip area needed for the resulting circuits are studies with two test sets.

INTRODUCTION

A digital circuit is tested by applying a set of input vectors and verifying the responses generated by comparing them against expected values. Recently, built-in self-testing (BIST) techniques [1-7] have gained importance as it increases the speed of testing. In BIST it is required that the circuit under test (CUT) and the associated testing circuitry, including test pattern generator (TPG) and test outputs evaluator (TOE), be on the same chip as shown in Figure 1. Many approaches to design of TPG have been proposed. They include Exhaustive testing [3], random pattern generation using Linear feedback shift register [4] and Store and Generate [5-7]. TOE may perform a simple comparison of generated and expected outputs for each input to CUT, or it may compare only the 'signatures' obtained by compacting outputs produced for a sequence of inputs. In general, TOE will consist of: (i) ERG - Expected Response Generator, (ii) RP-Response processor, and (iii) Comparator. ERG can be one or more stored signatures or a circuit which can generate the true responses for each test pattern. Most of the methods proposed for BIST treat TPG and TOE as un-correlated units. The problem of global minimization of area and time has not been addressed.

In this paper we study Store and Generate methods which require that the response produced by the application of each test vector be verified. Hence there is no need for RP. We propose a design procedure for TPG and ERG. The method is also applicable when TPG and ERG are combined into a single unit.

DESIGN OF TPG AND ERG

The test vectors can be stored in ROM on chip. TPG then reads each word of the ROM during testing. It may be economical to use a combination of smaller ROM and additional circuitry which can be used to generate the test vectors at time of testing. This was shown as a possibility for unordered random test vectors in [7]. The procedure may result in TPG generating additional (non-test) vectors. We give an alternate design procedure. In the following, let T be the given set of N test vectors, each n bit wide.

Procedure P - Design of TPG for unordered test set:

(P1) Rearrange T in ascending order by considering each vector as a decimal number.

(P2) Construct the set D, where $d_i = t_{i+1} - t_i, 0<i<N-1$.
Let dm be the maximum value in D.

(P3) Encode $d_i, 0<i<N-1$, in $\lceil \log_2(dm-1) \rceil$ bits and store in ROM.

(P4) Using t_1 as first test vectors, the remaining test vectors can be generated in succession by adding d_i and t_i to get t_{i+1}.

Design of TPG with ERG for unordered test set: The steps are same as in Procedure P except the vectors in T are n+m bits wide. It includes the m-bit outputs expected by applying n-bit test vector to CUT. Note, the choice of placing m bits in the n+m positions can be used to minimize the size of ROM storing the differences. Also, the number of test vectors N is not increased as may happen in the procedure of [7].

Procedure P was applied to two known test sets of the TI chip SN74181 which has 14 inputs and 8 outputs. Test set #1(#2) contains 46(27) vectors and requires ROM of 644(368) bits for storing inputs or 1012(594) bits for storing both inputs and outputs. Table below summarizes the effect of designing TPG and TPG with ERG. Thus, there is a 23% reduction in ROM size for test set #1. This approach to designing TPG and ERG requires that an adder be available on chip. Most modern processor chips do come with two adders, for example the Intel 8086/8088. Thus, savings in ROM area, comparable to the method of [7], can be achieved without need for additional circuitry. If Step P1 is removed then Procedure P can be applied for ordered test sets also. We found no savings in ROM size by this approach for ordered Test set #1 and #2. But we expect substantial savings for test sets which are larger subsets of the set of all vectors. Note that test set #1 has only 46 vectors out of 16,384 possible, whereas usual test sets are larger by two orders of magnitude.

		Optimal Difference	Width of ROM	No.of Words	ROM Size	% Change
TPG	Test Set #1	2,048	11	45	495	-23%
	Test Set #2	4,096	12	26	312	-12%
TPG with ERG	Test Set #1	524,288	19	45	855	-15%
	Test Set #2	1,048,578	20	26	520	-12%

CONCLUSIONS

We have proposed a new procedure to design test circuits for BIST. The method provides a viable alternative to existing methods. This is the first known method that takes into account the storage of both inputs and outputs needed for testing.

REFERENCES

[1] T.W.Williams,"VLSI Testing", IEEE Computer Magazine, pp.126-135, October 1984.

[2] "Built-in Test" - - Concepts and Techniques", IEEE Computer Society Publications, October 1983.

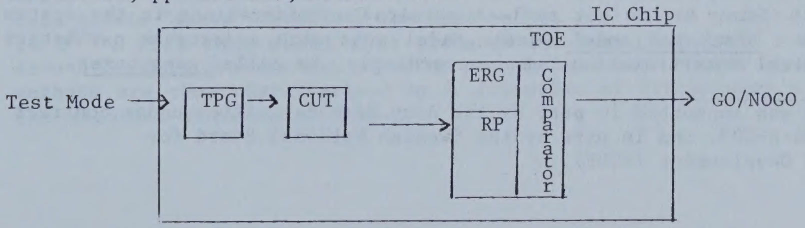
[3] E.J. McCluskey, "Build-In Verification Test", Digest, 1982 IEEE Test Conference, pp. 183-190, November 1982.

[4] B. Koenemann, J. Mucha, and G.Zwiehoff, "Build-in logic block observation techniques", Digest, 1979 IEEE Test Conference, pp. 37-41, November 1979.

[5] V.K. Agarwal and E.Cerny, "Store and generate built-in-testing approach", Proceedings of Fault Tolerant Computing Symposium-11 pp. 35-40 June 1981.

[6] M.E.Aboulhamid and E. Cerny, "A Class of Test Generators for built-in testing", IEEE Transaction on Computers, vol.C-32 pp. 957-959, October 1983.

[7] R.Dandapani, J.H.Patel and J.A. Abraham, "Digest,1984 IEEE Test Conference, pp. 315-319, October 1984.



SYSTEM IDENTIFICATION: AN EXPOSITORY SURVEY

LENNART LJUNG, Department of Electrical Engineering, Linköping University
S-581 83 Linköping, Sweden

At present with the Laboratory for Information and Decision
Systems, MIT, Cambridge, MA 02139.

Abstract

In this contribution a a brief overview of the system identification problem is given. The role of prediction and prediction error is emphasized, yielding a certain uniformity of the field. Frequency domain interpretations are also given to parametric, time-domain methods.

1 AN OUTLINE OF SYSTEM IDENTIFICATION

Inferring models from observations is really what science is about. System identification deals with the subfield that concerns the construction of mathematical models of dynamical systems, based on observed input-output data.

The identification procedure can in general terms be described as follows.

1. Collect input-output data from the process.
2. Settle for a set of candidate models.
3. Pick one particular member of the model set as the best representative, guided by the information in the data.

Let us give a conceptual discussion of each of these three steps. More details will follow below.

1. The data. The data is sometimes recorded during a specifically designed identification experiment. The objective then is to get as good information about the system as possible. Some methods to determine models require special input sequences, and thus specific experiments. In other cases data from normal operation of the system have to be used.

2. The set of models. A set of candidate models is obtained by specifying among which collection of models we are going to look for a suitable one. For graphic models this will typically be the set of all reasonably smooth curves. For analytic models suitable sets are usually obtained by letting certain parameters in the model descriptions range over a given space. We shall generally denote a set of models by M.

A model set whose parameters are basically viewed as vehicles for adjusting the fit to data, and do not reflect physical considerations in the system is called a black box model (set). Model sets with adjustable parameters with physical interpretations may, accordingly, be called grey boxes.

This work was supported in part by the Army Research office under contract DAAG-29-84-K-005, and in part by the Swedish National Board for Technical Development (STUF).

3. Picking a particular model, in M, guided by data. This is "the identification method". There is obviously a vast number of ways to select models. To give an idea about basic principles, it is useful to distinguish between methods for graphic models and for sets of analytic models.

Graphic time domain-models: Graphic time-domain models are basically impulse or step responses. Techniques to determine these are called transient analysis, see Rake (1980). The impulse response can also be obtained as the crosscorrelation function between the output and the input, when the input sequence is white noise, see, e.g. Godfrey (1980). Apparatus that performs such correlation analysis is commercially available.

Graphic frequency domain models: A direct way of determining the value of the frequency response function of a system at a given frequency, is to use a sinusodial input of that frequency, let the transient die out, and record the phase shift and amplitude change of the output sinusoid. This technique is known as frequency response analysis. See Rake (1980).

With more sophisticated data analysis one may, so to speak, apply all frequencies at the same time and sort them out afterwards by Fourier techniques. This gives a frequency response function

$$\hat{G}_N(e^{i\omega}) = Y_N(\omega)/U_N(\omega) \tag{1}$$

$$Y_N(\omega) = \frac{1}{\sqrt{N}} \sum_{t=1}^{N} y(t)e^{-i\omega t}$$

$$U_N(\omega) = \frac{1}{\sqrt{N}} \sum_{t=1}^{N} u(t)e^{-i\omega t}$$

where Y_N and U_N are the discrete Fourier transforms (DFT) of the output and input sequences using N data. When noise affects the systems (1) is usually a bad estimate ("the periodogram estimate"), since no noise reduction is obtained. Instead various smoothing filters are applied to (1):weighted averages over certain frequency windows are formed. Such techniques are known as spectral analysis and are further described in e.g. Jenkins and Watts (1968) and Brillinger (1981) The estimate can thus be written

$$\hat{G}_N(e^{i\omega}) = \int_{-\pi}^{\pi} W_\gamma(\zeta-\omega)\hat{G}_N(e^{i\zeta})d\zeta \tag{2}$$

where $W_\gamma(\zeta)$ is a weighting function that typically has its "mass" concentrated around $\zeta=0$. See also Ljung (1985).

Analytical models. Most modern identification methods deal with the estimation of analytical models, usually described in the time domain. More detailed descriptions of such methods will be given in Section 3. See also Goodwin and Payne (1977), Åström (1980) and Åström and Eykhoff (1971). The basic idea behind such methods is the following one: Let each of the candidate models "guess" (predict) the next output y(t) based on the information in Z^{t-1}. Pick that model that produces the best ("smallest") sequence of errors between guesses and actually recorded outputs. These identification methods are thus characterized by a criterion of fit between a model and the recorded sequence of data.

2 ANALYTIC TIME-DOMAIN MODELS

Describing dynamical systems in the time domain allows a considerable amount of freedom. Usually, differential (partial or ordinary) equations are used to describe the relationships between inputs and outputs. In discrete-time (sampled-data systems) difference equations are used instead. The question of how to describe properties of various disturbance signals also allows for several different possibilities. Here, we shall list a few typical choices, confining ourselves to the case of linear, discrete time models. We shall for easy notation use the sampling interval as the time unit, thus assuming it to be constant.

Linear difference equations. Let the relationship between the input sequence $\{u(t)\}$ and the output sequence $\{y(t)\}$ be described by

$$y(t)+a_1y(t-1)+....+a_ny(t-n)=b_1u(t-1)+....+b_mu(t-m) \tag{3}$$

Here the coefficients a_i and b_i are adjustable parameters. We shall generally denote the adjustable parameters by a vector Θ:

$$\Theta = (a_1...a_n\ b_1...b_m)^T. \tag{4}$$

If we introduce the vector of lagged inputs and outputs

$$\varphi(t)=\left(-y(t-1)...-y(t-n)u(t-1)...u(t-m)\right)^T, \tag{5}$$

the equation (3) can be rewritten in the more compact form

$$y(t) = \Theta^T\varphi(t). \tag{6}$$

In (3) or (6), the relationship between inputs and outputs is assumed to be exact. This may not be realistic in a number of cases. Then we may add a term $v(t)$ to (6):

$$y(t) = \Theta^T\varphi(t) + v(t) \tag{7}$$

that accounts for various noise sources and disturbances that affect the system, as well as for model inaccuracies. This term can be further modelled, typically by describing it as a stochastic process with certain properties. The simplest model of that kind is to assume $\{v(t)\}$ to be white noise, i.e. a sequence of independent random variables with zero mean values. However, many other possibilities exist. Among the most common models is the following one.

ARMAX models. If the term $\{v(t)\}$ in (7) is described as a moving average (MA) of white noise $\{e(t)\}$ we have a model

$$y(t)+a_1y(t-1)+...+a_ny(t-n)=b_1u(t-1)+...+$$

$$+b_mu(t-m)+e(t)+c_1e(t-1)+...+c_ne(t-n) \tag{8}$$

Such a model is known as an ARMAX model.

Output error models. Instead of adding the disturbance $v(t)$ to the equation as in (7), it can be added as an output measurement error:

$$y(t) = x(t) + v(t) \tag{9a}$$

$$x(t)+f_1x(t-1)+...+f_nx(t-n)=b_1u(t-1)+...+b_mu(t-m) \tag{9b}$$

Such models are often called output error models. The "noise-free output" $x(t)$ is here not available for measurement, but given (16b) it can be reconstructed from the input. We denote by $x(t,\Theta)$ the noisefree output that is constructed using the model parameters

$$\Theta = (f_1 \ldots f_n \; b_1 \; \ldots b_m)^T, \tag{10}$$

With

$$\varphi(t,\Theta)=\left(-x(t-1,\Theta)\ldots -x(t-n,\Theta) \; u(t-1)\ldots u(t-m)\right)^T, \tag{11}$$

(9) can then be rewritten as

$$y(t) = \Theta^T\varphi(t,\Theta) + v(t) \tag{12}$$

State-space models. A common way of describing stochastic, dynamical systems is to use state-space models. Then the relationship between input and output is described by

$$\begin{aligned} x(t+1) &= F(\Theta)x(t) + G(\Theta)u(t) + w(t) \\ y(t) &= H(\Theta)x(t) + e(t), \end{aligned} \tag{13}$$

where the noise sequences w and e are assumed to be independent at different time instants and have certain covariance matrices. Unknown, adjustable parameters Θ may enter the matrix elements in F, G and H in an arbitrary manner. These may, e.g., correspond to canonical parametrizations or to physical parameters in a time-continuous state space description that has been sampled to yield (13).

Models and Predictors

The list of potential models and model sets can be made long. For our purposes it is useful to extract the basic features of a model set, so as to allow for a treatment of model sets in general. First we introduce the following notation:

$\mathcal{M}(\Theta)$: a particular model, corresponding to the parameter value Θ

$\mathcal{M}$: a set of models: $\mathcal{M}=\{\mathcal{M}(\Theta) \mid \Theta \in D_{\mathcal{M}} \subset R^d\}$

Z^t: the set of measured input-output data up to time t:

$$Z^t=\{u(1),y(1),u(2),y(2)\ldots.u(t),y(t)\}$$

Similarly, u^t and y^t denote the input sequence and the output sequence, respectively, up to time t.

The various models that can be used for dynamical systems all represent different ways of thinking and representing relationships between measured signals. They have one feature in common, though. They all provide a rule for computing the next output or a prediction (or "guess") of the next output, given previous observations. This rule is, at time t, a function from Z^{t-1} to the space where $y(t)$ takes its values (R^p in general). It will also be parametrized in terms of the model parameter Θ. We shall use the notation

$$\hat{y}(t|\Theta) = g_M(\Theta;t,Z^{t-1}) \qquad (14)$$

for this mapping. The actual form of (14) will of course depend on the underlying model. For the linear difference equation (3)-(6) we will have

$$\hat{y}(t|\Theta) = \Theta^T\varphi(t) \qquad (15)$$

The same prediction or guess of the output y(t) will be used for the model (7) with disturbances, in case $\{v(t)\}$ is considered as "unpredictable" (like white noise). For the state space model (13) the predictor function is given by the Kalman filter. Then g_M is a linear function of past data.

The prediction $\hat{y}(t|\Theta)$ is computed from Z^{t-1} at time t-1. At time t the output y(t) is received. We can then evaluate how good the prediction was by computing

$$\varepsilon(t,\Theta) = y(t) - \hat{y}(t|\Theta). \qquad (16)$$

We shall call $\varepsilon(t,\Theta)$ the prediction error at time t, corresponding to model $M(\Theta)$. This term will be the generic name for general model sets. Depending on the character of the particular model set, other names like, e.g. the (generalized) equation error or output error may be used.

We can also adjoin an assumption about the stochastic properties of the prediction error to the model $M(\Theta)$:

"$M(\Theta)$: Assume that the prediction error $\varepsilon(t,\Theta)$ has the conditional (given Z^{t-1}) probability density function (p.d.f), $f(t,\Theta,x)$ (17)

$\left[\text{i.e. } P\left(\varepsilon(t,\Theta)\in B\right) = \int_{x\in B} f(t,\Theta,x)dx\right]$"

Notice that in (17) there is an implied assumption of independence of the prediction errors, for different t, since the p.d.f does not depend on Z^{t-1}. A predictor model (14) adjoined with a probabilistic assumption (17) we shall call a probabilistic model.

All the linear models listed here can be summarized in the form

$$y(t)=G(q,\Theta)u(t)+H(q,\Theta)e(t) \qquad (18)$$

with G and H as transfer functions of the shift operator q $[qu(t)=u(t+1)]$:

$$G(q,\Theta)=\sum_{k=1}^{\infty} g_k(\Theta)q^{-k}$$

$$H(q,\Theta)=1+\sum_{k=1}^{\infty} h_k(\Theta)q^{-k} \qquad (19)$$

[The state space model (13) must then first be converted to innovations form]

The one step ahead predictor for (18) takes the form

$$\hat{y}(t|\Theta)=H^{-1}(q,\Theta)G(q,\Theta)u(t)+ \left[1-H^{-1}(q,\Theta)\right]y(t) \qquad (20)$$

Guiding Principles behind Identification Methods

The problem now, is to decide upon how to use the information contained in Z^N to select a proper member $M(\Theta_N)$ in the model set, that is capable of "describing" the data. Formally speaking, we have to determine a mapping from Z^N to the set M:

$$Z^N \to M(\hat{\Theta}_N) \tag{21}$$

Now, how can such mapping be determined? We pointed out that the essence of a model of a dynamical system is its prediction aspect. It is then natural to judge the performance of a given model $M(\Theta^*)$ by evaluating the prediction errors, $\varepsilon(t,\Theta^*)$ given by (16). A guiding principle to form mappings (21), is thus the following one

"Based on Z^t, compute the prediction error $\varepsilon(t,\Theta)$ using (14) and (16). At time t=N select Θ_N so that the sequence of prediction errors $\varepsilon(t,\Theta_N)$, $t=1,\ldots,N$ becomes as small as possible".

The question is how to quantify what "small" should mean. Two approaches have been taken. These will be treated in the following two subsections.

Criterion minimization techniques

We introduce the scalar measure $\ell(t,\Theta,\varepsilon(t,\Theta))$ to evaluate "how large" the prediction error $\varepsilon(t,\Theta)$ is. Here ℓ is a mapping from $R\times R^d\times R^p$ to R, where $d=\dim\Theta$, $p=\dim y$. After having recorded data upp to time N a natural criterion of the validity of the model $M(\Theta)$ is

$$V_N(\Theta,z^N) = \frac{1}{N}\sum_{1}^{N} \ell(t,\Theta,\varepsilon(t,\Theta)) \tag{22}$$

This function is, for given Z^N, a well defined, scalar valued function of the model parameter Θ. The estimate at time N, i.e., Θ_N, is then determined by minimization of the function $V_N(\Theta,z^N)$. This gives us a large family of well known methods. Particular, "named" methods are obtained as special cases, corresponding to specific choices of model sets and criterion functions $\ell(t,\Theta,\varepsilon)$; and sometimes particular ways of minimizing (22).

The least squares method Choose $\ell(t,\Theta,\varepsilon)=|\varepsilon|^2$ and apply the criterion (22) to the difference equation model (7). Since the prediction is given by (15) we have the prediction error

$$\varepsilon(t,\Theta) = y(t) - \Theta^T\varphi(t).$$

The criterion function (22) thus becomes

$$V_N(\Theta,z^N) = \frac{1}{N}\sum_{1}^{N} |y(t)-\Theta^T\varphi(t)|^2, \tag{23}$$

which we recognize as the familiar least squares criterion. See, e.g. Strejc (1980). This function is quadratic in Θ, which is a consequence of the prediction being linear in Θ and the quadratic choice of criterion function.

To arrive at other specific functions ℓ, we could invoke, e.g. the maximum likelihood idea:

The maximum likelihood method For the probabilistic model (14), (17) the likelihood function can be found to be

$$\frac{1}{N} \log P\big(y(N), y(N-1), \ldots, y(1)\big) = \frac{1}{N} \sum_{1}^{N} \log f\big(t, \Theta, \varepsilon(t,\Theta)\big) \tag{24}$$

Maximizing the likelihood function is thus the same as minimizing the criterion (22) with

$$\ell(t,\Theta,\varepsilon) = -\log f(t,\Theta,\varepsilon) \tag{25}$$

We shall, following Ljung (1978), use the general term prediction error identification methods for the procedures we described in this section. See also Åström (1980).

Correlation techniques

Another way of expressing that the sequence $\{\varepsilon(t,\Theta)\}$ is small is to require that it be uncorrelated with a given sequence $\{\zeta(t)\}$. Let the vector $\zeta(t)$ represent some information that is available at time t-1.

$$\zeta(t) = \zeta(t, Z^{t-1}) \tag{26}$$

Sometimes, there is reason to consider a more sophisticated variant, where ζ itself may depend on the parameter Θ. (Some such cases will be discussed below):

$$\zeta(t) = \zeta(t, \Theta, Z^{t-1}) \tag{27}$$

The rationale for requiring $\varepsilon(t,\Theta)$ and $\zeta(t)$ to be uncorrelated is the following: The predictors $y(t|\Theta)$ should ideally utilize all available information at time t-1. Thus, the prediction errors $\varepsilon(t,\Theta)$ should be uncorrelated with such information. (If they are not, more information can be squeezed out from Z^{t-1}).

We thus determine $\hat{\Theta}_N$ as the solution of

$$f_N(\Theta, Z^N) = \frac{1}{N} \sum_{t=1}^{N} \varepsilon(t,\Theta)\zeta^T(t) = 0 \tag{28}$$

where, normally, the dimension of ζ is such that (28) gives a system of equations that is compatible with the dimension of Θ. When (28) is applied to the model (15) the well known instrumental variable method results. The vector ζ is then known as the instruments or the instrumental variables. See Söderström and Stoica (1983) for a further discussion of this method.

How to choose $\zeta(t)$? A way to make the estimate $\hat{\Theta}$ insensitive to the characteristics of the noise that affects the system is to choose ζ to depend on past inputs only

$$\zeta(t) = \zeta(t, u^{t-1}) \tag{29}$$

Then that contribution to $\varepsilon(t,\Theta)$ that has its origin in the noise will be uncorrelated with ζ for all Θ. Choices (29) are typical for the instrumental variable method. It turns out that such choices that give the best accuracy of the obtained estimates are obtained when u is filtered through filters associated with the true system (see Söderström and Stoica, 1983). We then have

$$\zeta(t) = \zeta(t, \Theta, u^{t-1}) \tag{30}$$

A frequency domain interpretation

For the general linear model (18)-(20) we have that the prediction error is

$$\varepsilon(t,\theta)=H^{-1}(q,\theta)[y(t)-G(q,\theta)u(t)]. \qquad (31)$$

By Parseval's relationship we then have for $\ell(\varepsilon)=\varepsilon^2$

$$V_N(\theta,Z^N)=\frac{1}{N}\sum_{t=1}^{N}\varepsilon^2(t,\theta)\approx\int_{-\pi}^{\pi}|Y_N(\omega)-G(e^{i\omega},\theta)U_N(\omega)|^2/|H(e^{i\omega},\theta)|^2 d\omega=$$

$$=\int_{-\pi}^{\pi}|\hat{G}_N(e^{i\omega})-G(e^{i\omega},\theta)|^2|U_N(\theta)|^2/|H(e^{i\omega},\theta)|^2 d\omega \qquad (32)$$

Here Y_N and U_N are the DFT:s of y^N and u^N and $\hat{G}_N$ is defined by (8).

Minimizing $V_N(\theta,Z^N)$ can thus be interpreted as fitting $G(e^{i\omega},\theta)$ to the Fourier transform estimate $\hat{G}_N(e^{i\omega})$ in a quadratic norm (32) with the "signal-to-noise ratio" $|U_N(\omega)|^2/|H(e^{i\omega},\theta)|^2$ as weighting function. This shows a conceptual relationship between the prediction error estimates and the spectral analysis estimate (2).

Computing the estimate

By (22) or (28) we have (implicitly) defined the estimate. It now "only" remains to compute it. This could be a laborious procedure, but is as such a standard problem in numerical analysis. See, e.g., Dennis and Schnabel (1983) for a recent treatment. In addition, the system identification field is characterized by a considerable amount of different algorithms, many of which can be interpreted as attempts to solve (22) or (28), perhaps utilizing specific structures that stem from the particular prediction function (14) employed. See Ljung (1982) for some aspects on this.

An important class of special minimization and solution methods is formed by procedures that are required to work sequentially with the data stream; recursive identification methods. See e.g. Ljung and Söderström (1983) for further discussions of this topic.

4 ASYMPTOTIC PROPERTIES OF IDENTIFIED MODELS

In this section we shall discuss the asymptotic analysis of the estimate $\hat{\theta}_N$ as $N\to\infty$. The analysis is a fairly technical problem, and it is beyond the scope of these notes to provide the theory itself. Instead, we shall quote the bottom line of the analysis, which has direct implications for the user choices to be discussed in Section 5. Furthermore, we shall here confine ourselves to the criterion minimization approach.

Convergence

Consider the criterion (22) with $\hat{\theta}_N$ as minimizing argument.

Suppose that the limit

$$\bar{V}(\theta) = \lim_{N\to\infty} E\, V_N(\theta,Z^N) \qquad (33)$$

exists, where "E" means expectation with respect to Z^N. Then, under weak regularity conditions,

$\hat{\Theta}_N$ converges w.p.1 to a minimum of $\bar{V}(\Theta)$ as N tends to infinity. (34)

This is true, whether or not the model set M is capable of a true description of the data. It is thus quite a general result. See Ljung (1978).

Let us now specialize to the following assumption

- $\ell(t,\Theta,\varepsilon)=\varepsilon^2$ (35a)
- The model set (29) is used (35b)
- The data Z^N has been generated by "the true system"

  S: $y(t)=G_0(q)u(t)+H_0(q)e(t)$ (35c)

  where $\{e(t)\}$ is a sequence of independent random variables with zero mean values and variance λ, and the spectrum of u, $\Phi_u(\omega)$, is nonzero.
- Introduce

  $D_T(S,M)=\{\Theta \mid G(e^{i\omega},\Theta)=G_0(e^{i\omega});H(e^{i\omega},\Theta)=H_0(e^{i\omega})$ for all $\omega\}$ (35d)

  and assume that $D_T(S,M)$ is nonempty.
- $e(t)$ is independent of y^{t-1} and u^{t-1} (35e)

It is now easy to verify that under (35) we have

$\hat{\Theta}_N \to D_T(S,M)$ w.p.1 as $N\to\infty$ (36)

which means that the estimates of the transfer functions are strongly consistent

Asymptotic distribution

Suppose now that $\hat{\Theta}_N$ converges to Θ^*, such that the matrix

$$\frac{d^2}{d\Theta^2}\bar{V}(\Theta^*)$$

is invertible. Then

$\sqrt{N}(\hat{\Theta}_N-\Theta^*) \in \text{As } N(0,P)$ (37)

which means that the random variable $\sqrt{N}(\hat{\Theta}_N-\Theta^*)$ converges in distribution to the normal distribution with zero mean and covariance matrix P. (Ljung and Caines, 1979). Here

$$P=[\bar{V}''(\Theta^*)]^{-1} \lim_{N\to\infty} N\cdot E\, V'_N(\Theta^*)[V'_N(\Theta^*)]^T \; [\bar{V}''(\Theta^*)]^{-1} \quad (38)$$

where prime and doubleprime, denotes differentiation once and twice, respectively, w.r.t Θ. If Θ^* is a value such that the assumption (17) is true for $\Theta=\Theta^*$, then it can be shown that the matrix P in (38) equals the Cramér-Rao lower bound for the covariance matrix of any (unbiased) estimator.

Let us evaluate the expression (38) under assumptions (35)

$$P=[E\psi(t,\Theta_0)\psi^T(t,\Theta_0)]^{-1}\psi(t,\Theta_0)\lambda\psi^T(t,\Theta_0)\times$$
$$\times[E\psi(t,\Theta_0)\psi^T(t,\Theta_0)]^{-1}=\lambda[E\psi(t,\Theta_0)\psi^T(t,\Theta_0)]^{-1} \qquad (39)$$

where $\psi(t,\Theta)$ is the gradient of the prediction

$$\psi(t,\Theta) = \frac{d}{d\Theta}\hat{y}(t|\Theta) \text{ (column vector of dimension = dim}\Theta\text{)} \qquad (40)$$

and Θ_0 is a value in $D_T(S,M)$.

This result has an interesting implication. Let

$$\bar{V}(\Theta) = E\,\varepsilon^2(t,\Theta) \qquad (41)$$

Then we can evaluate how good the model $M(\hat{\Theta}_N)$ is by calculating the value $\bar{V}(\hat{\Theta}_N)$.

Here $\hat{\Theta}_N$ is a random variable, and we may evaluate the expectation of $\bar{V}(\hat{\Theta}_N)$ w.r.t Θ_N. This gives, after some straightforward calculations

$$E\bar{V}(\hat{\Theta}_N)\sim E\,\bar{V}(\Theta_0)+\lambda\frac{\dim\Theta}{N}, \qquad (42)$$

using (40). Here "~" means asymptotically equal to.

The result (42) is remarkable in its generality. It tells us that the expected prediction error variance increases with the number of independent parameters in the model (once the model set is large enough to make $D_T(S,M)$ nonempty) irrespectively of where the parameters enter the model.

Frequency domain expressions

So far we have focused our attention on the parameter vector Θ. In many cases black-box models are employed and then the parameters are just means to describe a flexible set of models. For linear systems this means that we are more interested in the transfer function estimates

$$\hat{G}_N(e^{i\omega})=G(e^{i\omega},\hat{\Theta}_N)$$
$$\qquad (43)$$
$$\hat{H}_N(e^{i\omega})=H(e^{i\omega},\hat{\Theta}_N)$$

than in $\hat{\Theta}_N$ itself. It is therefore useful to characterize the properties of (43) directly.

First, with (35abc) we have

$$\bar{V}(\Theta)=E\varepsilon^2(t,\Theta)=\frac{1}{2\pi}\int_{-\pi}^{\pi}E\left|Y(\omega)-G(e^{i\omega},\Theta)U(\omega)\right|^2/\left|H(e^{i\omega},\Theta)\right|^2d\omega=$$

$$=\frac{1}{2\pi}\int_{-\pi}^{\pi}\left[\left|G_0(e^{i\omega})-G(e^{i\omega},\Theta)\right|^2\Phi_u(\omega)+\lambda\left|H_0(e^{i\omega})\right|^2\right]/\left|H(e^{i\omega},\Theta)\right|^2d\omega \qquad (44)$$

provided $\{u(t)\}$ and $\{e(t)\}$ are independent. Here $\Phi_u(\omega)$ is the input spectrum.

The expression (44) together with (34) now characterizes the limiting estimate.

We find that $\hat{G}$ is fitted to G_0 in a quadratic frequency-domain norm using the weighting function $\Phi_u(\omega)/|H(e^{i\omega},\hat{\ },\eta^*)|^2$ (the model's signal- to-noise ratio). Moreover the model spectrum $|H(e^{i\omega})|^2$ is fitted to the error spectrum. More consequences of the expression (44) are discussed in Wahlberg and Ljung (1986).

For the asymptotic distribution the following rather remarkable result can be shown: Consider the black-box model set (18) being of order n. Let the estimates be $G_N(e^{i\omega},n)$ and $H_N(e^{i\omega},n)$ and let the limit values as $N\to\infty$ be $G^*(e^{i\omega},n)$ and $H^*(e^{i\omega},n)$. Then

$$\sqrt{N}\begin{pmatrix}\hat{G}_N(e^{i\omega},n)-G^*(e^{i\omega},n)\\ \hat{H}_N(e^{i\omega},n)-H^*(e^{i\omega},n)\end{pmatrix} \in AsN\big(0,P_n(\omega)\big) \tag{45}$$

$$\lim_{n\to\infty}\frac{1}{n}P_n(\omega)=\Phi_v(\omega)\begin{bmatrix}\Phi_u(\omega) & \Phi_{ue}(\omega)\\ \Phi_{eu}(\omega) & \lambda\end{bmatrix}^{-1} \tag{46}$$

Here $\Phi_{ue}(\omega)$ is the cross spectrum between input $\{u(t)\}$ and noise $\{e(t)\}$ (which is zero if the system operates in open loop). $\Phi_v(\omega)$ is the additive noise spectrum: $\Phi_v(\omega)=\lambda|H_0(e^{i\omega})|^2$.

For a formal statement and proof of this result, see Ljung (1985).

If the identification experiment has been performed in open loop we thus have for the asymptotic variance of G:

$$\mathrm{Var}\,\hat{G}_N(e^{i\omega},n)\sim\frac{n}{N}\cdot\frac{\Phi_v(\omega)}{\Phi_u(\omega)} \tag{47}$$

5. SOME USER CHOICES

To determine a good identification procedure for a given situation means that the user has to make a number of choices. He or she has to select experimental conditions (which signals to measure, which inputs to use, sampling rates etc), the set of models, and the functions ℓ in (22) or ζ in (30). The asymptotic results quoted here, like the set D_c, into which the estimates converge (see (34)) and the asymptotic covariance matrix P (see (37) -(39)), are indeed functions of these listed design variables. Rational decisions of the design variables can thus be based on analysis of D_c and P. This analysis may be complex, but certain conclusions can be drawn from general considerations:

- The experimental conditions should be such that the predictions become sensitive w.r.t. interesting parameters (see (39)): If ψ is "large" then P will be "small".
- The model set should be parsimoneous when describing the system (use few parameters) according to (42);
- The choice of ℓ should be matched to the probabilistic properties of the prediction errors (optimal choice $\ell(\varepsilon)=-\log f(x)$, where $f(x)$ is the probability density function of $\varepsilon(t,\theta_0)$).

- Since, the best approximation of the system for the experimental conditions at hand is obtained asymptotically, the chosen conditions should resemble those for which the model is to be used.

In addition to this general advice, the asymptotic expressions (45)- (47) can be used to calculate some fairly explicit, asymptotic optimal, experiment designs. We examplify this for a simple special case:

Suppose that the design objective is to minimize a criterion

$$\int_{-\pi}^{\pi} \operatorname{Var} \hat{G}_N(e^{i\omega}) \cdot C(\omega)\, d\omega \tag{48}$$

with respect to the input spectrum subject to constrained input variance. With (47) this problem takes the following mathematical form:

$$\min_{\Phi_u(\cdot)} \int_{-\pi}^{\pi} \frac{\Phi_v(\omega)}{\Phi_u(\omega)} \cdot C(\omega) d\omega \tag{49}$$

$$\text{subject to } \int_{-\pi}^{\pi} \Phi_u(\omega) d\omega \leq \alpha$$

This problem has the solution

$$\Phi_u^{opt}(\omega) = \mu \cdot \sqrt{\Phi_v(\omega) \cdot C(\omega)} \tag{50}$$

where μ is a constant, adjusted so that equality is obtained in the constraint. See Ljung (1985) for more details.

6 CONCLUSIONS AND OUTLOOK

System identification has been a central research area in the control field over three decades. It has been characterized by quite diverse approaches and a multitude of suggested "methods". Potential users may find the area incoherent when consulting the literature. Recent years have brought more unification into the field and it may be expected that the different views on identification techniques are converging to a more coherent picture.

It is my feeling that the System Identification has reached a certain state of maturity: Several methods (packaged into nice interactive soft-ware products) have become everyday tools of analysis in many application areas, the asymptotic theory is fairly well understood and efficient algorithms are available. No doubt, several issues remain to be explored but it does not seem likely that they will overthrow the foundations outlined in this contribution.

One might hope that the insight and knowledge of statisticians would be more fully utilized and adapted to the system identification problem. This concerns in particular how to support the important choice of model set by inference from data. Good techniques are established for the model order choice in black-box type model sets, but insights for more difficult choices of model sets would be desirable.

On the other hand, the control-oriented identification literature has traditionally been more inclined to use advanced results and models, than, say, the signal processing literature. This is of course due to the latter's focus on simple and fast algorithms, resulting from rapid data streams. With the computer revolution one may expect an increased interest in sophisticated signal processing algorithms and perhaps a transfer of identification methods from the control area.

REFERENCES

K J Åström (1980). Maximum likelihood and prediction error methods. Automatica, Vol 16, pp 551-574.

K.J. Åström and P. Eykhoff (1971). System identification - a survey. Automatica, Vol 7, pp 123-167.

D.R. Brillinger (1981). Time series-Data Analysis and Theory, Holden- Day San Francisco.

J.E. Dennis and R. B. Schnabel (1983). Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Englewood Cliffs.

K. R. Godfrey (1980). Correlation methods, Automatica, vol 16, pp 527-534.

G.C. Goodwin and R. L. Payne (1977). Dynamic System Identification: Experiment Design and Data Analysis, Academic Press.

L. Ljung (1978). Convergence analysis of parametric identification method IEEE Trans on Automatic Control, Vol Ac-23, pp 770-783.

L. Ljung (1982). Identification methods, Proc IFAC Symp on Identification Washington DC, pp 11-18.

L. Ljung (1985) Asymptotic variance expressions for identified black-box transfer function models, IEEE Trans. Autom. Control, Vol AC-30, September 1985.

L. Ljung and P. E. Caines (1979). Asymptotic normality of prediction error estimation for approximate system models, Stochastics, Vol 3, pp 29-46.

L. Ljung and T. Söderström (1983). Theory and Practice of Recursive Identification, MIT Press, Cambridge, Mass.

H. Rake (1980). Step response and frequency response methods. Automatica, Vol 16, pp. 519-526.

T. Söderström and P. G. Stoica (1983). Instrumental Variable Methods for System Identification, Springer-Verlag, Berlin.

V. Strejc (1980). Least squares parameter estimation. Automatica, Vol 16, 535-550.

B. Wahlberg and L. Ljung (1986) Design variables for bias distribution in transfer function estimation, IEEE Trans. Autom. Control, Vol AC-31, February 1986.

ADAPTIVE CONTROL THEORY: THE LAST FIVE YEARS

Kumpati S. Narendra
Center for Systems Science, Yale University
New Haven, Ct. 06520.

Abstract: The paper attempts to review the theoretical work published in the adaptive control area since the resolution of the stability problem in the ideal case in 1980.

I. INTRODUCTION

Adaptive control theory is currently an active area of research. A large number of workers, armed with an arsenal of new methods, are attempting to solve a wide variety of problems in the area. For the first time since the exciting days in 1980 when the stability problem of the ideal case was resolved [1]-[4], the entire field appears to have a focus and the efforts of different researchers show signs of coming together. The author believes that the next few years will witness some significant breakthroughs in the field due to the efforts, not only of adaptive control theorists, but also of those in closely related areas such as robust linear control.

The practical importance of adaptive control has been stressed repeatedly by many authors. The digital technology needed for the implementation of adaptive control is available and, in many areas, the limits to design are dictated by the available theory. Adaptive control has been found to be economically attractive in many applications due to sustained, though small, improvements in efficiency. In many areas, particularly in process control, adaptive control has proven very effective and commercial self-tuning regulators are beginning to make their appearance in the market. Finally, in rapidly advancing areas of technology such as robotics and the control of flexible space structures, it is believed that adaptive control will be mandatory. It is therefore not surprising that the field is humming with activity at both the theoretical and practical levels and is attractive to the beginner and expert alike.

Adaptation is needed to cope with uncertainty regarding the plant to be controlled. Invariably, a controller structure is chosen which would suffice when there is complete knowledge regarding the plant. The parameters of the controller are then adjusted on-line using the input-output measurements of the entire system. In view of this, the overall system is nonlinear. Further, since a trajectory is generally prescribed which the output of the plant is required to follow, the system is also time-varying. Hence the design of adaptive systems can be considered as the design of special classes of nonlinear time-varying control systems. Like any control system, such systems are also required to be stable, fast and accurate. The emphasis at the present time is in understanding the mechanism of stability in such systems and in devising schemes which assure either local or global stability. Improvements in speed of response and accuracy are then achieved within the framework provided by the above schemes.

A review of the field of adaptive control can be based on the type of problems that are of interest, (e.g. bounded disturbances, time-varying parameters, use of a reduced order model to approximate the plant), the analytical methods used (e.g. Lyapunov methods, stability under persistent disturbances[5], averaging methods[6], growth rates of unbounded signals[7] within the adaptive loop and extensions of Lyapunov methods based on persistent excitation), whether local or global stability is required, and whether or not the arguments call for the persistent excitation of the reference inputs. In this paper, we shall broadly follow the tree

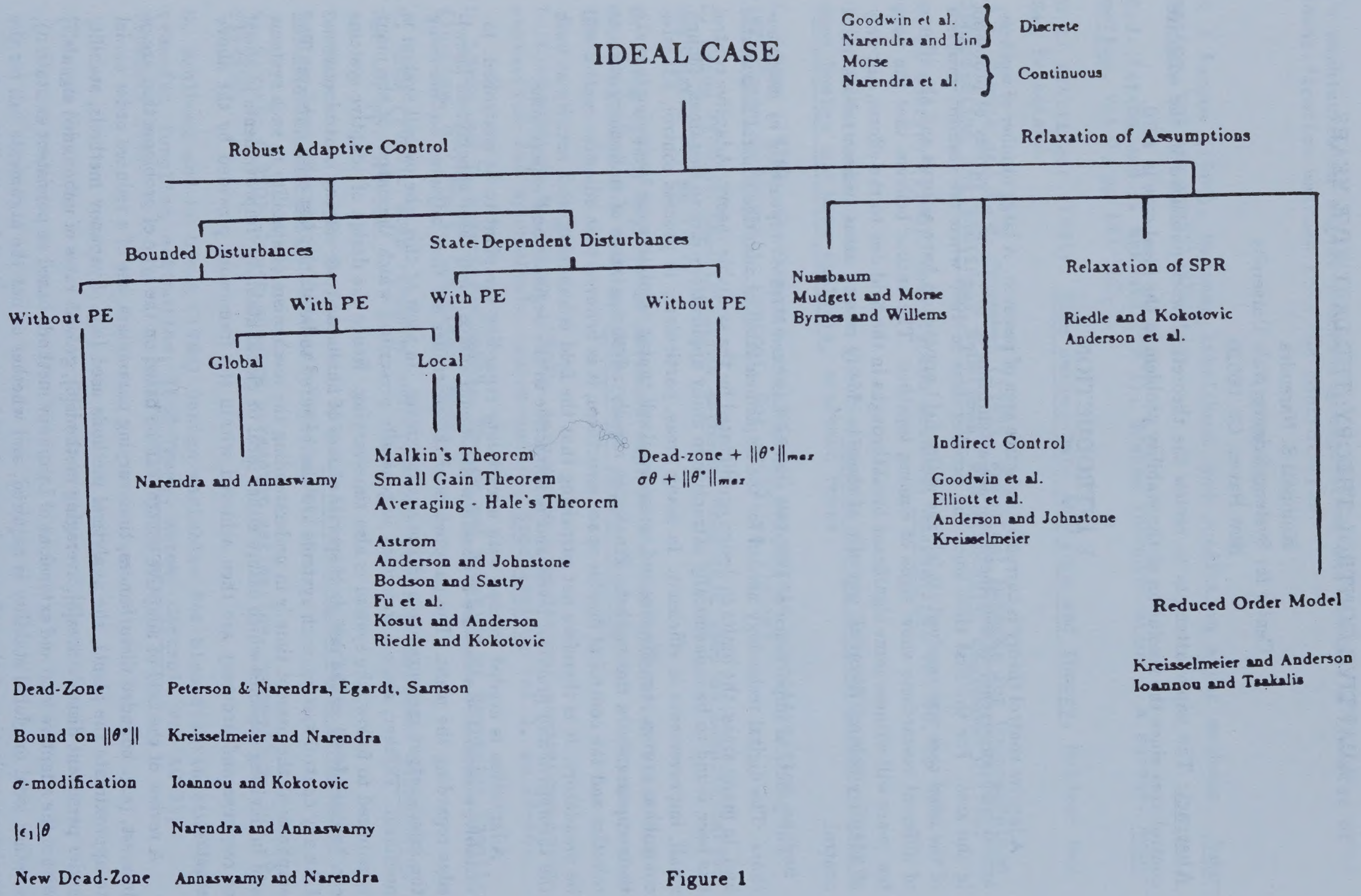

IDEAL CASE
Goodwin et al.
Narendra and Lin
Discrete
Morse
Narendra et al.
Continuous
Robust Adaptive Control
Relaxation of Assumptions
Bounded Disturbances
State-Dependent Disturbances
Without PE
With PE
Global
Local
Narendra and Annaswamy
With PE
Without PE
Malkin's Theorem
Small Gain Theorem
Averaging - Hale's Theorem
Astrom
Anderson and Johnstone
Bodson and Sastry
Fu et al.
Kosut and Anderson
Riedle and Kokotovic
Dead-zone + $\|\theta^*\|_{max}$
$\sigma\theta + \|\theta^*\|_{max}$
Nussbaum
Mudgett and Morse
Byrnes and Willems
Indirect Control
Goodwin et al.
Elliott et al.
Anderson and Johnstone
Kreisselmeier
Relaxation of SPR
Riedle and Kokotovic
Anderson et al.
Reduced Order Model
Kreisselmeier and Anderson
Ioannou and Tsakalis
Dead-Zone
Peterson & Narendra, Egardt, Samson
Bound on $\|\theta^*\|$
Kreisselmeier and Narendra
σ-modification
Ioannou and Kokotovic
$|\epsilon_1|\theta$
Narendra and Annaswamy
New Dead-Zone
Annaswamy and Narendra

Figure 1

structure indicated in Figure 1 to categorize the various developments since 1980. The paper deals entirely with deterministic, continuous time, single-input single-output systems, even though many of the concepts carry over to stochastic, discrete, and multivariable systems as well. The review is by no means exhaustive, and in view of space limitations, the emphasis is on concepts. The interested reader is referred to [8] by Narendra and Annaswamy for further analytical details.

II. THE IDEAL SYSTEM

The control problem whose stability was proven in 1980 forms the starting point for all the work reviewed in this paper and is now generally referred to as the ideal system. Figure 2 shows a block diagram of the model reference adaptive system, with $W_p(s)$ the unknown plant transfer function, (F, g) the filters used to generate the controller states, $\theta(t) \triangleq [\bar{\theta}^T(t), k(t)]^T$ the adaptive control parameter which is adjusted so that when $\theta(t) \equiv \theta^*$ (a desired control parameter vector), the closed loop transfer function matches the transfer function $W_m(s)$ of the reference model.

It is now well known that the sufficient conditions for choosing the model $W_m(s)$ as well as the controller structure to accomplish the above are knowledge of

(i) the sign of the high frequency gain k_p,
(ii) the relative degree n^* (number of poles - number of zeros),
(iii) an upper bound n on the order, of the plant transfer function $W_p(s)$ (I)
as well as the fact that
(iv) all zeros of $W_p(s)$ lie in the left half of the complex plane.

a. $n^* = 1$:

When the relative degree of the plant is unity, the model transfer function can be chosen to be strictly positive real and the stability problem is considerably simplified. In this case the adaptive law for adjusting the parameter vector $\theta(t)$ depends entirely on the output error $e_1(t)$ between plant and model outputs and the signals $\varpi(t)$ and $r(t)$ in the adaptive loop. In such a case the adaptive law is given by

$$\dot{\theta}(t) = \dot{\phi}(t) = -e_1(t)\omega(t) \tag{1}$$

where $\theta(t) = \theta^* + \phi(t)$, $\phi(t)$ is the parameter error vector and $\omega^T(t) = [\varpi^T(t),\ k(t)]$[9].

b. $n^* \geq 2$:

In contrast to the above, a more serious problem arises when the relative degree n^* is greater than unity. In such a case an augmented error is generated as shown in Figure 3 and the adaptive law takes the form

$$\dot{\theta}(t) = \dot{\phi}(t) = -\frac{\epsilon_1(t)\varsigma(t)}{1 + \varsigma^T(t)\varsigma(t)} \tag{2}$$

where $W_m(s)\omega \triangleq \varsigma$; $e_1 + \theta^T\varsigma - W_m(s)\theta^T\omega = \phi^T\varsigma = \epsilon_1$[10]. The adaptive laws (1) and (2) are referred to as the standard laws in the following sections. In both cases it can be shown that all the signals in the system are bounded and that $\lim_{t\to\infty} e_1(t) = 0$. Further, when the reference input is persistently exciting so that $\omega(t)$ in Figure 2 is persistently exciting, the parameter vector $\theta(t)$ converges to the desired value θ^*.

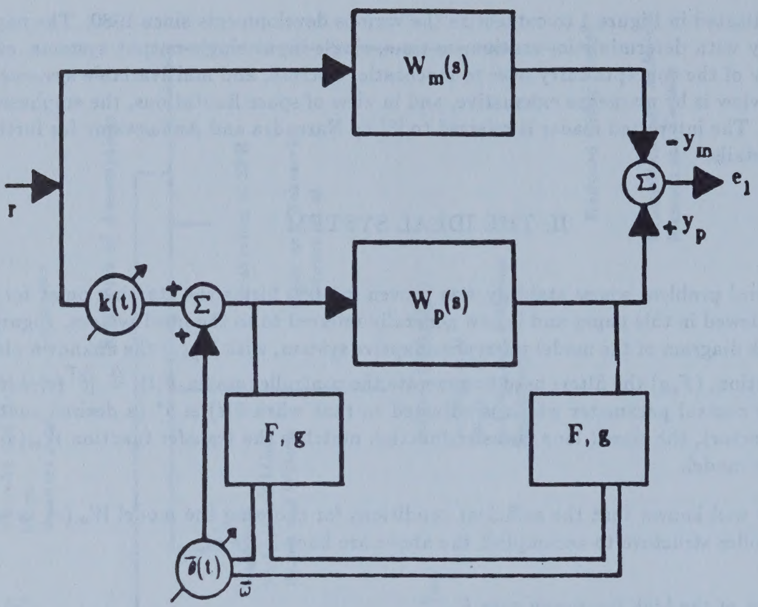

W_m(s)
-y_m
Σ
e_1
+y_p
r
k(t)
+
Σ
+
W_p(s)
F, g
F, g
θ̄(t)
ω̄

Figure 2: The Ideal Adaptive System: $n^* = 1$

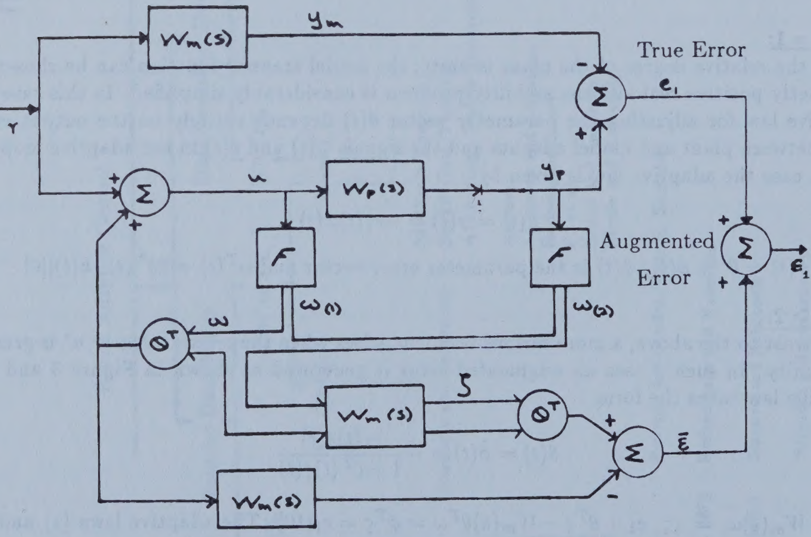

W_m(s)
y_m
True Error
-
Σ
e_1
+
r
+
Σ
+
u
W_p(s)
y_p
F
F
ω(1)
ω(2)
ω
θ^T
ζ
W_m(s)
θ^T
+
Σ
ε̄
-
W_m(s)
+
Augmented Error
Σ
ε_1
+

Figure 3: The Ideal Adaptive System: $n^* \geq 2$

Since many of the results derived in the later sections use arguments similar to that used in proving stability in the ideal case, we briefly outline the steps involved.

(i) Using a Lyapunov function it is first shown that ϕ is bounded. This implies that if the signals grow in an unbounded fashion then the signals $y_p, \omega^{(2)}, \omega$, and ς grow at the same rate [7].

(ii) It is next shown that $\dot{\phi} \in \mathcal{L}^2$ or equivalently $\epsilon_1 = \beta(t)\sqrt{1+\varsigma^T\varsigma}$ where $\beta \in \mathcal{L}^2$.

(iii) If $\dot{\phi} \in \mathcal{L}^2$, (ii) implies that $||\omega^{(2)}||$ and $||\omega(t)||$ grow at different rates which results in a contradiction. Hence all signals are bounded and $\lim_{t\to\infty} e_1(t) = \lim_{t\to\infty} \epsilon_1(t) = 0$.

(iv) Finally when $\omega(t)$ is persistently exciting, $\lim_{t\to\infty} \phi(t) = 0$ so that $\theta(t) \to \theta^*$.

c. Convergence of Parameters:

By condition (iv) the control parameter vector $\theta(t)$ converges to the true value θ^* when $\omega(t)$ is persistently exciting. However, it is clear that the persistent excitation of $\omega(t)$ cannot be demanded directly since it is a dependent variable and the output of a nonlinear system. This fact was recognized by many authors [11]-[13] who later derived the convergence of $\theta(t)$ based on the persistent excitation of the reference input $r(t)$. Based on these results it can be stated that $\lim_{t\to\infty} \theta(t) = \theta^*$ if $r(t)$ contains n distinct frequencies (or n spectral lines according to [13]).

III. ROBUST ADAPTIVE CONTROL IN THE PRESENCE OF BOUNDED DISTURBANCES

The various approaches that have been suggested to address the problem of bounded external disturbances can be divided into two categories. In the first category are included methods in which the adaptive law is suitably modified, without using arguments involving the persistent excitation of the reference input. These are by definition global in character and assure the boundedness of all the signals for all initial conditions. The methods in the second category include both local and global results and are based on persistent excitation arguments.

1. Modifications In Adaptive Law:

When bounded external disturbances are present, the choice of the reference model as well as the controller structure are identical to those in the ideal case. The overall system equations in this case are given by [14]

$$\dot{x} = Ax + b(\phi^T\omega + r + \bar{\nu}(t)); \qquad y_p = h^T x$$

where $x^T = [x_p^T,\ \omega^T]$, $h^T(sI-A)^{-1}b = W_m(s)$ and $\bar{\nu}(t)$ is an equivalent input disturbance. Assuming that $n^* \geq 2$, four adaptive laws are currently known for this problem. The error equation which relates the augmented error ϵ_1 to the parameter error ϕ is given by

$$\phi^T(t)\varsigma(t) + \nu(t) = \epsilon_1(t) \tag{3}$$

where $\nu(t) = W_m(s)\bar{\nu}(t)$.

a) Dead-Zone: In [14] Peterson and Narendra suggested the use of a dead-zone in the adaptive law so that the control vector is adjusted only when the augmented error is greater than a known upper bound ν_0 of the disturbance $\nu(t)$. In [14] the adaptive law takes the form

$$\dot{\phi} = \begin{array}{ll} = -\frac{\epsilon_1\varsigma}{1+\varsigma^T\varsigma} & |\epsilon_1| \geq \nu_0 + \delta \\ = 0 & |\epsilon_1| \leq \nu_0 + \delta \end{array} \tag{4}$$

where δ is an arbitrary positive constant. To prove boundedness of all signals, as in the ideal case, the boundedness of ϕ is first demonstrated which implies that signals $\omega^{(2)}$ and ω grow at the same rate if they grow in an unbounded fashion. It is later shown that $\dot{\phi} \in \mathcal{L}^2$ which implies that $\|\omega^{(2)}(t)\|$ and $\|\omega(t)\|$ grow at different rates leading to a contradiction. Hence boundedness of all signals is assured.

In contrast to the ideal case the output error does not converge to zero but merely to a value less than ν_0. It can also be shown that this error lies outside the dead-zone only for a finite time so that in most practical systems adaptation takes place only for a finite time. The use of a dead-zone can be considered as a conservative approach wherein adaptation is carried out only when the magnitude of the error guarantees that the parameter vector will be adjusted in the right direction. It is also obvious that a bound ν_0 on the disturbance is assumed to be known in this case.

The use of a dead-zone was also suggested by Egardt [27], and Samson [28] for the discrete case.

b) Bound on $\|\theta^*\|$: An alternate approach to the robustness problem in the presence of bounded disturbances was taken by Kreisselmeier and Narendra in [15]. Instead of stopping adaptation inside a "dead-zone", it is assumed in [15] that the desired vector θ^* has a norm less than a specified value $\|\theta^*\|_{max}$. Hence the search procedure for the control parameter vector can be confined essentially to the set $S : \stackrel{\Delta}{=} \{\theta| \, \|\theta\| \leq \|\theta^*\|_{max}\}$. The adaptive law in this case is given by

$$\begin{aligned} \dot{\phi} &= -\tfrac{\epsilon_1 \varsigma}{1+\xi^T\xi} && \text{if } \|\theta\| < \|\theta^*\|_{max} \\ &= -\tfrac{\epsilon_1 \varsigma}{1+\xi^T\xi} - \theta\left(1 - \tfrac{\|\theta\|}{\|\theta^*\|_{max}}\right)^2 && \text{elsewhere} \end{aligned} \tag{5}$$

where $\xi^T \stackrel{\Delta}{=} [\omega^T, \varsigma^T]$. In [15] it is shown that the above adaptive law results in the boundedness of all the signals in the system. While the boundedness of the parameter error vector $\phi(t)$ follows directly from the adaptie law, it cannot be shown that $\dot{\phi} \in \mathcal{L}^2$ since $\phi(t)$ does not converge as $t \rightarrow \infty$. However the proof of boundedness is qualitatively similar to that given in the previous case.

Assuming that the signals in the adaptive loop grow in an unbounded fashion, $\lim_{t\to\infty} \sup_{\tau \leq t} \|\xi(\tau)\| = \infty$ and unbounded sequences $\{t_i\}, \{a_i\}$ exist such that

$$\|\xi(t)\| \geq a_i \; \forall t \in [t_i, t_i + a_i]. \tag{6}$$

From the adaptive law (5) it also follows that

$$\int_{t_i}^{t_i+a_i} \frac{\epsilon_1^2}{1+\xi^T\xi} dt \leq \Delta \tag{7}$$

where Δ is a positive constant. Condition (7) plays the same role as that played by $\dot{\phi} \in \mathcal{L}^2$ in the ideal case and leads to a contradiction if it is assumed that all signals grow in an unbounded fashion.

In contrast to case a, an upper bound on the magnitude of the disturbance is not needed in this case but an upper bound on $\|\theta^*\|$ is needed instead to constrain the region of search in the parameter space.

Since the output error rarely tends to zero, the conditions discussed in this section generally prevail in most adaptive control problems. Hence the convergence arguments used in this case are found to be applicable in many other cases as well.

c) The σ-modification scheme [16]: In approaches III-a and III-b, certain prior information in the form of the bound on the disturbance or the desired parameter vector is assumed to implement the adaptive laws. In contrast to this, a scheme suggested by Ioannou and Kokotovic [16] assures boundedness of all signals in the system, without any such assumptions. The motivation for this method may be described briefly as follows: If $V(e,\phi)$ is a quadratic Lyapunov function candidate, the time-derivative $\dot{V}(e,\phi)$ has the general form $-e^TQe+e^T\alpha\bar{\nu}$ where $Q=Q^T>0, \alpha$ is a constant vector and $\bar{\nu}$ is a bounded disturbance. Very little can be concluded from this regarding the boundedness of e and ϕ and accounts for the modifications suggested in [14] and [15]. In [16], an additional term $-\sigma\theta$ is used in the adaptive law, as a result of which $V(e,\phi)$ becomes negative definite outside a bounded region in the (e,ϕ) space. From this it follows that all signals in the system are bounded.

While the use of the additional term $-\sigma\theta$ in the adaptive law is certainly a novel one, the approach suffers to some extent due to the very presence of this term. Since $-\sigma\theta=\sigma(\theta^*+\phi)$ where θ^* is the desired parameter vector, the error equations can never be u.a.s. if the plant that is being controlled is not stable to begin with (i.e. if $\theta^*\neq 0$). Further, the magnitude of the bias term is proportional to θ^* and hence, qualitatively, to the degree of adaptation needed. Many of these weaknesses can be compensated for by the proper choice of σ but in no case can the bias be entirely eliminated if prior information regarding the plant is limited to that available in the ideal case. The possibility of modifying the adaptive law when a bound on $\|\theta\|$ is known, as in [15], was recently suggested by Ioannou and Tsakalis [18] for the general case when $n^*\geq 2$. In such a case, the adaptive law does not significantly differ from that suggested in [15].

d) A New Adaptive Law: Recently a new adaptive law motivated by that given in [16], was proposed by Narendra and Annaswamy in [17]. Like the adaptive law in [16], it does not require prior information regarding the bounds on the unknown parameters or the magnitude of the disturbance. Further in the absence of any disturbances, the error equations are uniformly asymptotically stable when the reference input is sufficiently persistently exciting. This makes this scheme particularly attractive for tracking problems in the presence of disturbances. To the authors' knowledge, this scheme appears to be superior to all the schemes currently known in this category since it requires no additional prior information for its implementation, results in bounded solutions without requiring persistent excitation, and enjoys the property of u.a.s. in the ideal case when the reference input is sufficiently persistently exciting.

The new adaptive law proposed is given by

$$\dot{\phi}(t)=-\frac{\epsilon_1\varsigma}{1+\xi^T\xi}-\gamma\frac{|\epsilon_1|\theta}{1+\xi^T\xi} \qquad \gamma>0. \tag{8}$$

It is shown in [17] that the adaptive law (8) leads to

(i) boundedness of all signals in the system in the ideal case,

(ii) u.a.s. of equation (8) when ς is sufficiently persistently exciting, and

(iii) boundedness of all signals in the system when a bounded disturbance $\nu(t)$ is present.

The proofs of (i) and (iii) follows along the same lines as in [15] and the reader is referred to [8] for further details.

2. Robustness with Persistent Excitation:

The importance of persistent excitation in adaptive identification and control is now well realized. It was studied in the late seventies extensively by Yuan and Wonham [19], Morgan and Narendra[20,21], Anderson [22], and Sondhi and Mitra [23] and was shown to result in the exponential stability of certain linear time-varying systems. Since exponential convergence and its concomitant property of robustness are important in practical applications, the concept has been used extensively in the study of robust adaptive systems in the last five years. While results based on persistent excitation have been reported for the control of systems with bounded external disturbances, time-varying parameters and unmodeled dynamics, we shall confine our attention in this section to the first two and relegate the discussion of the last problem to section V.

a) Local Stability: Extensive work has been done in the Soviet Union in the past concerning stability under persistent disturbances and is found to be most relevant to the analysis of many adaptive systems. In particular, the concept of total stability and Malkin's theorem which assures such stability feature quite often in discussions of robustness. For example, in robust adaptive control we are interested in deducing the properrties of the solutions of a perturbed system (S_p) from the solutions of an unperturbed system (S). These can be described by the differential equations

$$\dot{x} = f(x,t) \qquad (S); \qquad\qquad \dot{z} = f(z,t) + g(z,t) \qquad (S_p) \tag{9}$$

Definition (Total Stability)[24]: The equilibrium state $x = 0$ of (S) is totally stable if for every $\epsilon > 0$ two positive numbers $\delta_1(\epsilon)$ and $\delta_2(\epsilon)$ exist such that every solution $p(t, x_0, t_0)$ of (S_p) satisfies $||p(t, x_0, t_0)|| < \epsilon, \;\; t \geq t_0$ provided $||x_0|| < \delta_1$ and $||g(x,t)|| < \delta_2$.

In the Russian literature this is also referred to as stability under persistent disturbances. The fact that the u.a.s. of the unperturbed system implies total stability was shown by Malkin [5] and is frequently used to prove robustness of adaptive systems in the presence of sufficiently small perturbations.

Theorem: If the equilibrium state of (S) is uniformly asymptotically stable, then it is totally stable.

In adaptive systems the unperturbed system is known to be u.a.s. when the reference input is persistently exicting. Hence, Malkin's theorem can be interpreted in terms of the initial conditions as well as the bounds on the perturbations. It appears possible to treat nonlinearities, disturbances and plant parameters in a unified fashion using such an approach [25]. In this manner, robust adaptive control of time-varying plants has been treated in [26]. However, the magnitude constraint on the initial condition is relaxed in this paper at the expense of a stronger condition on $g(z,t)$.

b) Global Stability: In practical systems we are interested in the uniform boundedness of the solutions in the presence of perturbations as well as in the magnitudes of these bounds. This leads to the concept of practical stability [29]. Recently, Narendra and Annaswamy [30] gave such a result which, in the author's opinion, will have important implications in all aspects of robust adaptive control. Given the error equations

$$\begin{aligned} \dot{e} &= Ae + b(\phi^T \omega + \nu); \qquad e_1 = h^T e \\ \dot{\phi} &= -e_1 \omega \end{aligned} \tag{10}$$

where ν is a bounded perturbation with $|\nu(t)| \leq \nu_0$. The problem is to determine the conditions on the reference input $r(t)$ which will assure global boundedness of the solutions of (10). Since with $\nu(t) \equiv 0$ and ω persistently exciting, the system is exponentially stable, the condition for the boundedness of solutions of (10) can be expressed in terms of the persistent excitation of ω. However, the latter cannot be assured a priori since ω is a dependent variable of the system. Hence if $\omega = \omega^* + Ce$ (where ω^* corresponds to the signal ω in the reference model), the condition can be expressed in terms of the persistent excitation of ω^*. Rewriting equation (10), we have the set of nonlineaar time-varying differential equations

$$\begin{aligned} \dot{e} &= Ae + b\phi^T[\omega^* + Ce] + \nu; \qquad e_1 = h^T e \\ \dot{\phi} &= -e_1[\omega^* + Ce]. \end{aligned} \tag{11}$$

In [30] it is shown that if ω^* is persistently exciting and has a degree of persistent excitation ϵ_0, then equation (11) will have bounded solutions provided

$$\epsilon_0 \geq c_1 \nu_0$$

where c_1 is a constant dependent on the system parameters.

From the above result it follows that given any bounded disturbance, the system can be made to have bounded solutions by merely increasing the degree of persistent excitation of the reference input. This is in contrast to the local results given earlier, which merely assure the existence of sufficiently small initial conditions and perturbations when the refrence input is persistently exciting to achieve the same objective.

The results in [30] apply to the case of a system with relative degree $n^* = 1$. Work is currently in progress to extend the results to the general case where $n^* \geq 2$.

3. Hybrid Adaptive Control:

Recently Narendra, Khalifa and Annaswamy [31] presented several new adaptive algorithms which can be applied to continuous or discrete systems. The common feature of such algorithms is that control parameters are adjusted at rates significantly slower than the rates at which the system operates. In discrete systems this corresponds to infrequent adjustment of the control parameters based on the information collected over several instants of time. The resulting algorithms involve reduced computational effort while making the system more robust under perturbations [12].

If the parameter error and the output error are denoted as $\phi(.)$ and $e_1(.)$ respectively, the error equations for a discrete adaptive control problem can be expressed as

$$\begin{aligned} \phi^T(k)\omega(l) &= e_1(l) \quad k,l \in N, l \in [kT,(k+1)T) \quad \text{(ideal case)} \\ \phi^T(k)\omega(l) + \nu(l) &= e_1(l) \qquad \text{(with bounded perturbations)} \end{aligned}$$

Throughout the interval $[kT,(k+1)T]$, the error e_1 and the signal $\omega(.)$ are measured while the parameter vector θ (and hence ϕ) remains constant. At the end of the interval, using one of the adaptive laws suggested in [31], given by

$$\Delta\theta(k) = \Delta\phi(k) = -\frac{1}{T}\sum_{l=kT}^{(k+1)T} \frac{e_1(l)\omega(l)}{1+\omega^T(l)\omega(l)}$$

the parameter vector $\theta(k)$ is updated. The resulting equation has the form

$$\phi(k+1) = [I - R(k)]\phi(k) + S(k)$$

where $R(k) = \frac{1}{T}\sum_{l=kT}^{(k+1)T} \frac{\omega(l)\omega^T(l)}{1+\omega^T(l)\omega(l)}$, and $S(k) = \frac{1}{T}\sum_{l=kT}^{(k+1)T} \frac{\omega(l)\nu(l)}{1+\omega^T(l)\omega(l)}$.

Qualitatively, the above control can be said to improve the degree of persistent excitation while decreasing the effective magnitude of the disturbance. Hence, by the results of the previous section, the robustness of the overall system is significantly improved. In the author's opinion hybrid adaptive control is preferable in all situations over instantaneous adjustment of the control parameters.

IV. RELAXATION OF ASSUMPTIONS ON PLANT TRANSFER FUNCTION

The four assumptions in (I) given in section II were needed to obtain a complete solution to the stability problem in the ideal case. Since these assumptions are rarely met in practice, efforts were made during the past few years to determine how these could be relazed. Such efforts can be broadly classified under two categories. In the first category can be included all those approaches which treat violations of the above assumptions as perturbations on the system. These can be considered as contributions to robust adaptive control. In contrast to this the second category includes parallel efforts in which the necessity of the assumptions in (I) is questioned. In this section we describe briefly the advances reported on this aspect of the adaptive control problem.

1. Sign of k_p:

Consider the differential equation

$$\begin{aligned} \dot{x} &= ax + \lambda f(x,y) \\ \dot{y} &= g(x,y) \end{aligned} \tag{12}$$

A conjecture was made by Morse [32] that functions f and g cannot be found which would stabilize the above system when the sign of λ is unknown. Nussbaum [33] showed that while the conjecture is true for the case when f and g are rational functions of x and y, it is not true in general. In particular, he constructed the following example in which, by the proper choice of f and g, he showed that $\lim_{t\to\infty} x(t) = 0$ while y remains bounded:

$$\begin{aligned} \dot{x} &= ax + \lambda x(y^2+1)h(y) \\ \dot{y} &= x(y^2+1). \end{aligned} \tag{13}$$

This is accomplished by choosing $h(y)$ as an even, differentiable function satisfying

$$\sup_{u>y_0} \int_{y_0}^{u} h(s)ds = +\infty \qquad \text{and} \quad \inf_{u>y_0} \int_{y_0}^{u} h(s)ds = -\infty.$$

The above result has had a major impact on the researchers in the adaptive control area concerned about relaxing the assumptions on the plant transfer function. Using the above approach, Mudgett and Morse [34] have shown that the sign of the high frequency gain is not needed to stabilize the adaptive system. The corresponding result for discrete systems has been derived by Lee and Narendra [35]. In [36] Willems and Byrnes consider a plant of arbitrary order with a relative degree unity, all zeros in the left half plane, and with the sign of the high frequency gain unknown. In this case, it follows that either a large positive gain or a large negative gain in the feedback path will stabilize the overall system. By a suitable parametrization, it is shown in [36] that the system can be stabilized. This problems indicates that at least in special cases the bound on the order of the system may not be needed.

For a long time it was assumed that the assumptions in (I) are not only sufficient but also necessary to stabilize an unknown plant. The contribution of Nussbaum and the results that followed exemplify that good theory may lead to answers which are counter intuitive.

2. Indirect Control:

Indirect adaptive control methods rely on the estimation of the plant parameters to adjust the control parameters. One of the main aims of those using indirect control has been the elimination of the requirement that the zeros of the plant lie in the open left half plane. Since matching plant and model outputs asymptotically with zeros in the right half plane could result in unstable pole-zero cancellations, the latter formulations merely require pole-placement of the adaptive loop. It is argued that if the parameter estimates of the plant converge to their true values, stability of the adaptive system could be assured by pole-placement of the adaptive loop. It is argued that if the parameter estimates of the plant converge to their true values, stability of the adaptive system could be assured by pole-placement even if condition (iv) in (I) is not satisfied.

One of the problems that arises in Indirect Control is that the estimates of the plant parameters may be such that the numerator and denominator polynomials of the estimated transfer function are not coprime. In such a case the computed control can become unbounded. Several efforts were made to prevent the occurrence of such situations. In [37] Anderson and Johnstone demonstrate that by modifying the adaptive law when the estimates lie in a certain region, parameter convergence can be assured. They require prior information regarding the plant parameters (in the form of an under bound on the Sylvester determinant) to implement the adaptive law.

Efforts by Goodwin et al. [38] and Elliott et al. [39] use block processing in which the plant parameters are estimated over blocks of time when the control parameters are constant. In such a case, persistently exciting reference inputs assure the convergence of the parameter estimates to their true values and hence boundedness of all the signals. While this procedure may not be practical, it demonstrates that the assumption (iv) is not necessary to control a plant adaptively.

V. ADAPTIVE CONTROL USING REDUCED ORDER MODEL

In the opinion of the author (and one which is shared by many others in the field) the problem of adaptively controlling an unknown plant using a reduced order controller is the most important problem in the field at the present time. The ramifications of this problem are felt in almost all open questions in the field and even a partial resolution of this problem will have a significant impact on both theory and practice.

The plant is assumed to consist of two transfer functions $W_p(s)$ and $\overline{W}_p(s)$ in parallel. $W_p(s)$ is the modeled part and prior information concerning it is assumed to be known so that a controller can be designed to match a given reference model as in the ideal case. $\overline{W}_p(s)$ on the other hand represents the dynamics of the plant that is unmodeled. The aim of adaptive control is then to determine an adaptive law for adjusting the control parameter vector so that all the signals in the adaptive loop are bounded and the error between plant and model outputs satisfies some performance criterion.

The output η of $\overline{W}_p(s)$ in the above description of the problem can be considered as a state-dependent disturbance on the plant. Hence it cannot be assumed to be bounded a priori so that the methods outlined in section III cannot be directly applied. As in earlier cases, the approaches that have been suggested for this problem can also be divided into two categories. In the first, the arguments used to assure boundedness depend crucially on persistent excitation, while in the second, persistent excitation of the reference signal is not called for.

1. Approaches Using Persistent Excitation:

In recent years, the largest number of publications have been in this area of adaptive control. Only a few representative papers are discussed in this section to indicate the approaches used and the results obtained.

The discussion of total stability and Malkin's theorem in section III shows that they can be used to consider the reduced order model problem as well. Such an approach has been taken by Kosut and Johnson [40] and Kosut and Anderson [41]. Other mathematical methods used include averaging as well as the application of the small gain theorem. Invariably all the methods assume persistent excitation of the reference input so that the ideal system is u.a.s. and all the results obtained are local in character, assuring the boundedness of all the signals when the perturbations are small in some sense.

a) *Method of Averaging:* This is a standard method which has been used extensively in the study of nonlinear oscillations. Originally suggested by Krylov and Bogoliuboff [42] it was later developed by Bogoliuboff and Mitropolsky [43] and more recently by Hale [6] and Arnold [44]. In this method, assuming that the right hand side of a differential equation contains a small parameter ϵ, an approximate solution can be obtained by somewhat intuitive considerations. Thus if

$$\dot{x} = \epsilon f(t, x, \epsilon) \qquad x(0) = x_0 \tag{14}$$

then $\dot{x}$ is small and x varies slowly. Hence it is quite logical to expect that the rapidly varying terms in f do not effect the slow variation of x in the long term. By a process of averaging, the nonautonomous differential equation (14) can be approximated by an autonomous differential equation in the variable x_{av} which represents the average value of x. This makes the problem mathematically tractable. In the adaptive control problem, it has been shown that the proof of stability is based on the slow variations of the control parametrs as compared to other state variables. Hence the averaging method is found to particularly applicable to such a problem. This was first suggested by Åström[45] in an effort to analyze instability resulting from unmodeled dynamics.

The method of averaging has been applied by Fu et al. [46], Kokotovic et al. [47], and Kosut et al. [48] to adaptive control in the presence of unmodeled dynamics. Fu et al. [46] provide an excellent introduction to the averaging method and apply it to some standard identification and control problems. In [47] Kokotovic et al conclude that the stability of the adaptive system is critically dependent on the frequency content of the excitation of ω in relation to the frequency response of a related time-invariant transfer function $H(s)$ (which is the closed loop transfer function of the plant together with the controller when $\theta(t) \equiv \theta^*$). Kosut et al. [48] extend the averaging method to the case where the rapidly varying signals ω belong to the class $\mathcal{P}_{[0,\infty)}$ defined in [19]. They demonstrate that for a sufficiently small ϵ, the overall system will have bounded solutions if the dominant excitation acts in the frequency range where $H(s)$ is strictly positive real (SPR).

From the foregoing discussion it is clear that many of the difficulties encountered are due to the fact that the transfer function $H(s)$ is not SPR so that there is a possibility of the control parameter vector being adjusted in the wrong direction. An alternate approach to the reduced order model problem has been suggested by Anderson et al. in [49] where it is shown, using the small gain theorem [50], that if the reference input is persistently exciting and has low frequencies, $H(s)$ is not SPR at high frequencies, and ϵ is sufficiently small, then the overall system will have bounded solutions.

2. Approaches Using Modifications of the Adaptive Law:

While the results presented in the previous section are of interest in obtaining insights regarding the instability mechanism in adaptive systems, our main interest is in obtaining conditions which assure boundedness of solutions for arbitrary initial conditions. Two such results have been recently reported.

a) Varying Dead-Zone and bound on $||\theta^*||$: It was shown in section III that if a bound ν_0 on the disturbance is known, a dead-zone can be used in the adaptive law to assure the boundedness of the parameter vector. The same argument would also carry over to the case where ν_0 is varying with time and a time-varying dead-zone is used. In [51] it is assumed that the stable unmodeled part of the plant is known at least to the extent that an instantaneous bound on its output $\eta(t)$ can be computed on-line. Further it is also assumed that a bound on the magnitude of the desired control vector θ^* is also known. Assuming that the unmodeled part has a gain μ, it is shown that a combination of the two techniques developed for bounded disturbances can be extended to the reduced order problem by choosing μ to be sufficiently small.

b) σ-modification and bound on $||\theta^*||$: In [18] a combination of schemes III-b and III-c is used to derive a similar result. The plant is assumed to be described by the differential equation

$$\begin{aligned} \dot{x}_0 &= Ax_0 + b_0 u \\ \mu\dot{\eta} &= A_f\eta + \mu b_f u \\ y_0 &= c_0^T x_0 + \mu c_f \eta \end{aligned} \tag{15}$$

so that the unmodeled output η is a small but rapidly varying signal if μ is small. The controller is designed, as in the earlier sections, to stabilize the modeled part using a combination of schemes III-b and III-c and it is shown that the overall system has bounded solutions if μ is sufficiently small.

The above two schemes have revealed that globally bounded solutions to the adaptive control problem can be derived using suitable adaptive laws under the assumption that the transfer function of the unmodeled part is stable and has a sufficiently small gain. Since the procedures in the two cases follow closely those presented in section III for bounded disturbances, it now appears possible to derive a unified approach for the treatment for the reduced order model problem. Work is currently in progress in this area.

VI. CONCLUSIONS

The developments in adaptive control theory, since the resolution of the global stability problem in 1980 of the ideal case, have been treated briefly in the preceding sections. The emphasis throughout the paper is on conditions which assure the boundedness of all the signals in the adaptive system when different types of perturbations are present.

From the above results it can be concluded that robust adaptive control under various perturbations can be achieved either by modifying the adaptive laws or by requiring the reference input to be persistently exciting. In fact the author and his coworkers believe that the above two approaches may be used interchangeably in most situations. The attempts to relax the assumptions on the plant transfer function discussed in section IV lead us to conclude that none of the conditions (i)-(iv) in (I) may be necessary for successful adaptive control. This, in turn, makes the adaptive control problem much less restrictive than was believed in 1980.

It is well known that the adaptive control problem in the ideal case can be divided into an algebraic part and an analytic part. The former is concerned with the existence of a time-invariant controller which can achieve the desired objective. The latter deals with the asymptotic convergence of the time-varying controller, using an adaptive law, to the time-invariant controller. A similar division can also be made in all the robust adaptive control problems described in this paper. The relaxation of the assumptions on the plant transfer function now makes the algebraic part, particularly for the reduced order model problem, closer in spirit to the problems currently being discussed in the area of robust linear control. The discussions in the paper also reveal that there is a wide choice of approaches while addressing the analytic part of the problem. It is the confluence of these different ideas that leads the author to believe that some significant developments can be anticipated in the near future in the resolution of the reduced order model problem.

ACKNOWLEDGMENTS

The work reported here was supported by the National Science Foundation under Grant ECS-8300223. The author would like to thank his co-worker Dr. Annaswamy for numerous discussions on all aspects of the adaptive control problem considered in this paper.

References

[1] K.S. Narendra, Y.H. Lin and L.S. Valavani, "Stable Adaptive Controller Design - Part II:Proof of Stability," *IEEE Trans. on Autom. Control*, AC-25, pp. 440-448, June 1980.

[2] A.S. Morse, "Global Stability of Parameter Adaptive Control Systems," *IEEE Trans. on Autom. Control*, AC-25, pp. 433-439, June 1980.

[3] K.S. Narendra and Y.H. Lin, "Stable Discrete Adaptive Control," *IEEE Trans. on Autom. Control*, AC-25, pp. 456-461, June 1980.

[4] G.C. Goodwin, P.J. Ramadge and P.E. Caines, "Discrete time Multivariable Adaptive Control," *IEEE Transactions on Automatic Control*, vol. 25, pp. 449-456, June 1980.

[5] I.G. Malkin, "Theory of Stability of Motion," Tech. Report Tr. 3352, U.S. Atomic Energy Commission, English ed. 1958.

[6] J.K. Hale, Ordinary Differential Equations, Kreiger, Molaban, Florida, 1980.

[7] K.S. Narendra, A.M. Annaswamy and R.P. Singh, "A General Approach to the Stability Analysis of Adaptive Systems," *International Journal of Control*, vol. 41, No. 1, 1985.

[8] K.S. Narendra and A.M. Annaswamy, "Robust Adaptive Control," Adaptive and Learning Systems: Theory and Applications, Ed: K.S. Narendra, Plenum Press, New York, 1985.

[9] K.S. Narendra and L.S. Valavani, "Stable Adaptive Controller Design-Direct Control," *IEEE Trans. on Autom. Control*, AC-23, pp.570-583, Aug. 1978.

[10] K.S. Narendra and Y.H. Lin, "Design of Stable Model Reference Adaptive Controllers," Applications of Adaptive Control, pp. 69-130, Academic Press, New York, 1980.

[11] S. Dasgupta, B.D.O. Anderson and A.C. Tsoi, "Input Conditions for Continuous Time Adaptive Systems Problems," Tech. Report, Australian National University, 1983.

[12] K.S. Narendra and A. M. Annaswamy, "Persistent Excitation and Robust Adaptive Systems," Tech. Report No. 8302, Center for Systems Science, Yale University (Also in the Proceedings of the Third Yale Workshop on Applications of Adaptive Systems Theory, Yale University, June 1983.).

[13] S. Boyd and S.S. Sastry, "On Parameter Convergence in Adaptive Control," *Systems and Control Letters"*, vol. 3, pp. 311-319, 1983.

[14] B.B. Peterson and K.S. Narendra, "Bounded Error Adaptive Control," *IEEE Trans. on Autom. Control*, AC-27, pp. 1161-1168, Dec. 1982.

[15] G. Kreisselmeier and K.S. Narendra, "Stable Model Reference Adaptive Control in the Presence of Bounded Disturbances," *IEEE Trans. on Autom. Control*, AC-27, pp. 1169-1175, Dec. 1982.

[16] P.A. Ioannou and P.V. Kokotovic, "Adaptive Systems with Reduced Models," Springer-Verlag, New York, 1983.

[17] K.S. Narendra and A.M. Annaswamy, "A New Adaptive Law for Robust Adaptive Control Without Persistent Excitation," Tech. Report No. 8510, Center for Systems Science, Yale University, New Haven, September 1985.

[18] P.A. Ioannou and K. Tsakalis, "A Robust Direct Adaptive Controller," Tech. Report 85-07-1, Univ. of Southern California, May 1985.

[19] J.S.-C Yuan and W.M. Wonham, "Probing Signals for Model Reference Identification," *IEEE Transactions on Automatic Control*, AC-22, pp. 530-538, August 1977.

[20] A.P. Morgan and K.S. Narendra, "On the Uniform Asymptotic Stability of Certain Linear Nonautonomous Differential Equations" *SIAM Journal of Control and Optimization*, Vol. 15, pp. 5-24, Jan. 1977.

[21] A.P. Morgan and K.S. Narendra, "On the Stability of Nonautonomous Differential Equations $\dot{x} = [A + B(t)]x$ with Skew-Symmetric Matrix $B(t)$" *SIAM Journal of Control and Optimization*, Vol. 15, pp. 163-176, Jan. 1977.

[22] B.D.O. Anderson, "Exponential Stability of Linear Equations Arising in Adaptive Identification," *IEEE Transactions on Automatic Control*, AC-22, pp. 83-88, Feb. 1977.

[23] M. M. Sondi and D. Mitra, "New Results on the Performance of a Well-Known Class of Adaptive Filters," Proc. IEEE, Vol. 64, pp. 1583-1597, 1976.

[24] W. Hahn, Theory and Application of Liapunov's Direct Method, Prentice-Hall International Series in Applied Mathematics, Englewood Cliffs, New Jersey, 1963.

[25] M. Bodson and S.S. Sastry, "Small Signal I/O Stability of Nonlinear Control Systems - Applications to the Robustness of a MRAC Scheme," Tech. Report, UC at Berkeley, September 1985.

[26] B.D.O. Anderson and R.M. Johnstone, "Adaptive Systems and Time-varying Plants," *International Journal of Control*, vol. 37, No. 2, pp. 367-377, Feb. 1983.

[27] B. Egardt, Stability of Adaptive Controllers, Springer- Verlag, Berlin, 1979.

[28] C. Samson, "Stability Analysis of Adaptively Controlled System Subject to Bounded Disturbances,", *Automatica*, vol. 19, No. 1, pp. 81-86, 1983.

[29] J.P. LaSalle and S. Lefschetz, Stability by Liapunov's Direct Method, Academic Press, New York, 1961.

[30] K.S. Narendra and A.M. Annaswamy, "Robust Adaptive Control in the Presence of Bounded Disturbances," Tech. Report 8406, Center for Systems Science, Yale University, New Haven, Ct., 1984. (Also presented at the 1984 ACC conference, San Diego, 1984.); To appear in the *IEEE Transactions on Automatic Control.*

[31] K.S. Narendra, I.H. Khalifa and A.M. Annaswamy, "Error Models for Stable Hybrid Adaptive Systems - Part II," *IEEE Transactions on Automatic Control*, AC-30, No. 4, pp. 339-347, April 1985.

[32] A.S. Morse, "Recent Problems in Parameter Adaptive Control," Proc. CNRS colloquirm on Development and Utilization of Mathematical Models in Automatic Control," Belle-Isle, France, Sept. 1982.

[33] R.D. Nussbaum, "Some Remarks on a Conjecture in Parameter Adaptive Control," *Systems and Control Letters*, vol. 3, pp. 243-246, 1983.

[34] D.R. Mudgett and A.S. Morse, "Adaptive Stabilization of Linear Systems with Unknown High-Frequency Gains," *IEEE Transactions on Automatic Control*, AC-30, pp. 549-554, June 1985.

[35] T.H. Lee and K.S. Narendra, "Stable Discrete Adaptive Control with Unknown High Frequency Gain," Tech. Report No. 8408, Center for Systems Science, Yale University, New Haven, Ct., Nov. 1984.(Also submitted to the IEEE Trans. on Autom. Control).

[36] J. Willems and C. Byrnes, "Global Adaptive Stabilization in the Absence of Information on the Sign of the High Frequency Gain," *Proc. INRIA Conf. on Analysis and Optimization of Systems, Springer Lecture Notes in Control and Information Sciences*, vol. 62, pp. 49-57, June 1984.

[37] B.D.O. Anderson and R.M. Johnstone, "Global Adaptive Pole Positioning," *IEEE Transactions on Automatic Control*, AC-30, pp. 11-22, Jan. 1985.

[38] G.C. Goodwin and E.K. Teoh, "Persistency of Excitation in the Presence of Possibly Unbounded Signals," *IEEE Transactions on Automatic Control*, AC-30, pp. 595-597, June 1985.

[39] H. Elliott, R. Cristi and M. Das, "Global Stability of Adaptive Pole Placement Algorithms," *IEEE Transactions on Automatic Control*, AC-30, pp. 348-356, April 1985.

[40] R.L. Kosut and C.R. Johnson, Jr., "An Input-Output View of Robustness in Adaptive Control," Special Issue on Adaptive Control, *Automatica*, September 1984.

[41] R.L. Kosut and B.D.O. Anderson, "Robust Adaptive Control: Conditions for Local Stability," *Proceedings of the 23rd IEEE CDC*, Las Vegas, 1984.

[42] A.N. Krylov and N.N. Bogoliuboff, Introduction to Nonlinear Mechanics, (English Translation), Princeton University Press, Princeton, New Jersey, 1943.

[43] N.N. Bogoliuboff and Yu. A. Mitropolskii, Asymptotic Methods in the Theory of Nonlinear Oscillators, Gordon & Breach, New York, 1961.

[44] V.I. Arnold, Geometric Methods in the Theory of Differential Equations, Springer-Verlag, New York, 1982.

[45] K.J. Åström, "Analysis of Rohrs Counter-example to Adaptive Control," *Proceedings of the 22nd IEEE CDC*, San Antonio, Texas, pp. 982-987, Dec. 1983.

[46] L.-C. Fu, M. Bodson and S.S. Sastry, "New Stability Theorems for Averaging and Their Applications to the Convergence Analysis of Adaptive Identification and Control Schemes," Tech. Report UCB/ERL M85/21, UC at Berkeley, March 1985.

[47] P. Kokotovic, B. Riedle and L. Praly, "On a Stability Criterion for Continuous Slow Adaptation," *Systems & Control Letters*, vol. 6, pp. 7-14, June 1985.

[48] R.L. Kosut, B.D.O. Anderson and I. Mareels, "Stability Theory for Adaptive Systems: Methods of Averaging and Persistent Excitation," Preprint, 1985.

[49] B.D.O. Anderson, R.R. Bitmead, C.R. Johnson, Jr., and R.L. Kosut, "Stability Theorems for the Relaxation of the Strictly Positive Real Condition in Hyperstable Adaptive Schemes," *Proceedings of the 23rd IEEE CDC*, Las Vegas, 1984.

[50] C.A. Desoer and M. Vidyasagar, Feedback Systems: Input-Output Properties, Academic Press, 1985.

[51] G. Kreisselmeier and B.D.O. Anderson, "Robust Model Reference Adaptive Control," DFVLR Tech. Report, Oberfaffenhofen, W. Germany, 1984.

Area-Time Complexity of Rectangular Matrix Multiplication (Extended Abstract)

King F. Pang[1] Abbas El Gamal[2]
Information Systems Laboratory
Electrical Engineering Department
Stanford University
Stanford, CA 94305 USA

ABSTRACT

The power of a recently developed two-dimensional multiple circuit partitioning technique is demonstrated by proving the tightest known AT^2 lower bound for rectangular matrix multiplication. The proof method is extended to three-dimensional partitions and the subsequent information transfer requirement gives AT lower bounds. For rectangular matrix multiplication, the AT lower bound proved is tight up to logarithmic factors. This result is uniformly stronger than that obtained from combining the corresponding AT^2 and A bounds. Tight area lower bound is also proved for the problem through its bisection information flow across time.

---

[1] Work partially supported under DARPA Contract MDA 903-79-C-0680 and NSF Grant ECS83-00988. Current Address: Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304.

[2] Work partially supported under DARPA Contract MDA 903-79-C-0680.

1. Introduction

The bisection information-flow proof method [Thom80] has been employed to derive tight AT^2 lower bounds for a variety of problems, such as DFT, integer multiplication and convolution. However, there still are a number of problems for which the lower bounds obtained are orders of magnitude smaller than the best known upper bounds. Examples are encoding and decoding functions for error correcting codes, as well as the class of l-independent functions [Grig76]. In order to obtain better AT^2 lower bounds for these problems, two-dimensional (2D) multiple circuit partitions have been introduced [EGP84]. The idea is to partition the circuit into many blocks and show that at least one block must have a certain number of inputs (resp. outputs), no more than a certain number of outputs (resp. inputs) and a small perimeter. The information flow requirement across the boundary of this block is then used to lower bound AT^2.

In this paper, we extend 2D partitions so that cuts can potentially be made over all three dimensions of the circuit, including time. By bounding the subsequent information flow, AT lower bounds are obtained. We call such a partition technique three-dimensional (3D) partitions. For a given problem, the information flow requirement of a block in a 2D partition is *a priori* a lower bound of that in a 3D partition. Hence, by bounding the former, both AT^2 and AT lower bounds can be proved.

We apply these proof methods to obtain the tightest known AT and AT^2 lower bounds for rectangular matrix multiplication. We show that any VLSI circuit that multiplies an $m \times n$ binary matrix and an $n \times p$ binary matrix must satisfy $AT^2 = \Omega(mnpq)$, where $q = \min(m,n,p)$, and $AT = \Omega(mnp)$. The combinatorial result used in proving these two lower bounds also allows us to obtain a tight bisection information flow lower bound for the problem. This gives the achievable lower bound $A = \Omega(mnp/r)$, where $r = \max(m,n,p)$. Notice that the AT lower bound is much stronger than that obtained from combining the corresponding AT^2 and A bounds. This vividly demonstrates the strength of 3D multiple partitions.

The remainder of this paper is organized as follows: We briefly introduce 2D and 3D multiple partitions in Section 2. The relations between the information flow requirements and the corresponding area-time lower bounds are outlined. These general results are applied to rectangular matrix multiplication in Section 3, where the AT^2, AT and A lower bound proofs will be sketched.

2. Multiple Partitions

The VLSI circuit model adopted in this paper is similar the *Grid Model* described in Chapter 2 of [Ullm83], to which we refer those readers interested in the details. To specify its behavior over time, we use a three-dimensional grid to model a circuit during the computation of one instance of a problem. As 2D and 3D multiple circuit partitions, as well as the information-flow proof methods are described in detail in [EGP84] and [Pang85], we only give the pertinent definitions and state without proof the main results in this section [†]. We use f to denote a generic problem with n inputs and m outputs.

Definition 1: Let $\mathcal{U}$ be the 3D grid representation of a circuit which computes an instance of f. A *$(3,I,s)$-partition* (resp. *$(3,O,s)$-partition*) of $\mathcal{U}$ is one which cut it, possibly over all three dimensions, into s blocks each with n/s inputs (resp. m/s outputs). □

Definition 2: A *$(2,I,s)$-partition* (resp. *$(2,O,s)$-partition*) of a circuit computing f is one which cuts across its plane, into s blocks each with n/s inputs (resp. m/s outputs). □

Lemma 1: If a circuit which computes f is $(3,I,s)$-partitioned (resp. $(2,I,s)$-partitioned), then there must exist a block which has:

(i) n/s inputs;

(ii) at most $2m/s$ outputs;

(iii) boundary size $O((AT/s)^{2/3})$ (resp. $O(T\sqrt{A/s})$). □

There is a corresponding lemma for $(3,O,s)$- and $(2,O,s)$-partitions, by interchanging the roles of inputs and outputs. The following definition classifies functions according to their 3D and 2D information flows:

Definition 3: A problem f is said to have *$(3,s)$-flow* (resp. *$(2,s)$-flow*) I bits if for any circuit to correctly compute all its outputs, one of the following has to be satisfied:

(1) In an $(3,I,s)$-partition (resp. $(2,I,s)$-partition), any block with no more than $2m/s$ outputs has communication requirement of at least I bits; or

(2) In an $(3,O,s)$-partitioned (resp. $(2,O,s)$-partition), any block with no more than $2n/s$ inputs has communication requirement of at least I bits. □

The relations between information flow and area-time lower bounds are stated in the following:

Theorem 1: Let f be a problem with $(2,s)$-flow I bits, then any circuit which computes f must satisfy $AT = \Omega(sI^{3/2})$ and $AT^2 = \Omega(sI^2)$. □

3. Rectangular Matrix Multiplication

We restrict the discussions in this section to binary matrices. However, all results are extendible to an arbitrary ring straightforwardly.

Definition 4: The *(m,n,p)-matrix multiplication* problem is to compute $Z = X \cdot Y$, where X and Y are $m \times n$ and $n \times p$ binary matrix respectively. □

[†] Note added in proof: *Tessellation*, a technique for multiple partitions of VLSI circuits similar (but *not* identical) to ours, have been reported in [Angl82] and [BP86].

The VLSI Complexity of (m,n,p)-matrix multiplication was previously investigated in [Sava81] who proved $AT^2 = \Omega(m^2p^2)$ for $(a-n)(b-n) < n^2/2$, where $a = \max(m,n)$ and $b = \max(n,p)$. The bound $AT^2 = \Omega(n^2 \min(m,p)^2)$, for $m,p > n$ has also been reported [Lott84]. Using multiple partitions, a much stronger AT^2 lower bound, tight to within logarithmic factors, will be proved in Theorem 2. Nearly tight AT and tight A lower bounds will also be derived.

Incidentally, the AT^2 lower bound in Theorem 2 has been obtained independently by D. Angluin [Angl82]. The proof there also uses 2D circuit partition, but only a bound on the number of inputs and the boundary size of blocks are imposed. Unfortunately, only a sketch of the argument was given. It is therefore difficult to assess the similarities between her proof and ours. However, we believe that the nearly tight AT bound and the tight area bound in this paper are new.

The following lemma forms the core of the information flow lower bound. Its proof can be found in the Appendix.

Lemma 3: Let Q be a $k \times l$ binary matrix, where $k \geq n$, $l \geq n$ and $0 < c \leq 1/8$. If $\|Q\| = n^2$, $\|Q(i,*)\| > 0$ for all i and $\|Q(*,j)\| > 0$ for all j, then one of the following must be satisfied:

(1) There exist cn rows $Q(j_i,*)$ such that $\sum_{i=1}^{cn} \|Q(j_i,*)\| \geq cn^2$.

(2) There exists a family of disjoint sets of columns of Q, $\{Q^{(t)} : 1 \leq t \leq s\}$, where $s \leq n$. Let $S_t = \bigvee_{Q(*,l) \in Q^{(t)}} Q(*,l)$, then the following are satisfied:

(a) $\|S_t\| \geq cn$ and $\sum_{Q(*,l) \in Q^{(t)}} \|Q(*,l)\| \leq 3\|S_t\|$ for $1 \leq t \leq s$.

(b) $\sum_{t=1}^{s} \|S_t\| \geq cn^2$. □

Lemma 4: If $n = \min(m,n,p)$, then the (m,n,p)-matrix multiplication problem has $(2,pm/n^2)$-flow $\Omega(n^2)$ bits.

Proof: We first argue that there is no loss of generality in assuming $n \leq c_1 m$ for a constant $0 < c_1 \leq c/8 \leq 1/64$. If $c_1 m \leq n \leq m$, by considering the multiplication of an arbitrary $m \times c_1 m$ submatrix of X with an arbitrary $c_1 m \times p$ submatrix of Y, the same asymptotic lower bound can be proved. Similarly, we can assume that $n \leq c_1 p$.

Next, perform a $(2,O,pm/n^2)$-partition. There are exactly n^2 output bits in each block. Moreover, there must exist a block with no more than $4n^3/p \leq cn^2/2$ input bits from X and $4n^3/m \leq cn^2/2$ input bits from Y. Consider such a block. Partition the elements of X into two sets, according to whether they are inside the block or not. The binary matrix $\mathcal{X}$ is then defined as

$$\mathcal{X}(i,j) = \begin{cases} 1 & \text{if } X(i,j) \text{ is inside the block,} \\ 0 & \text{otherwise.} \end{cases}$$

The binary matrices $\mathcal{Y}$ and $\mathcal{Z}$ are similarly constructed. We shall show that regardless of the locations of the n^2 ones in $\mathcal{Z}$, $\Omega(n^2)$ input bits have to flow in from the outside to compute them. The proof is divided into two cases:

Case 1: There exists rows $Z(l_t,*)$, $1 \le t \le cn$, such that $\sum_{t=1}^{cn} \|\mathfrak{z}(l_t,*)\| \ge cn^2$: There must also exist rows $Y(k_t,*)$, $1 \le t \le cn$, such that $\sum_{t=1}^{cn} \|\mathfrak{y}(k_t,*)\| \le c^2n^2/2$. By setting X to the appropriate permutation matrix, $Y(k_t,*)$ is mapped into $Z(l_t,*)$ for $1 \le t \le cn$. To compute these output elements, at least cn^2 bits are required. But no more than $c^2n^2/2$ of them are available in the block. Hence at least $(c-c^2)n^2/2 = \Omega(n^2)$ bits have to flow into the block to compute its outputs.

Case 2: The condition of Case 1 is not satisfied: Consider $\dot{\mathfrak{z}}$, a submatrix of $\mathfrak{z}$ formed by removing its all-zero rows and columns. Let Z^* be the corresponding submatrix of Z. Clearly, $\dot{\mathfrak{z}}$ satisfies the condition of Lemma 3. Hence, there must exist a family of disjoint sets of columns of $\dot{\mathfrak{z}}$, $\{\dot{\mathfrak{z}}^{(t)} : 1 \le t \le s \le n\}$, such that

(a) $\|S_t\| \triangleq \| \bigvee_{\dot{\mathfrak{z}}(*,j) \in \dot{\mathfrak{z}}^{(t)}} \dot{\mathfrak{z}}(*,j)\| \ge cn$ and $\sum_{\dot{\mathfrak{z}}(*,j) \in \dot{\mathfrak{z}}^{(t)}} \|\dot{\mathfrak{z}}(*,j)\| \le 3\|S_t\|$ for all $1 \le t \le n$,

(b) $\sum_{t=1}^{s} \|S_t\| \ge cn^2$.

Let $\{Z^{(t)} : 1 \le t \le s\}$ be the corresponding set of columns of Z^*. Set Y to the appropriate permutation matrix such that for $t = 1,\dots,s$, $X(*,t)$ is mapped into *every* column of $Z^{(t)}$. For each t, let

$$m_t = \sum_{i=1}^{m} 1(\dot{\mathfrak{z}}(i,j) = 1 \text{ for some } Z^*(*,j) \in Z^{(t)}).$$

From Lemma 3, $\sum_{t=1}^{s} m_t \ge cn^2$. In other words, at least cn^2 input elements from X are required for the computation of Z^*. Since no more than $cn^2/2$ of these are available, at least $(c-(c/2))n^2 = \Omega(n^2)$ bits must flow into the block. □

Lemma 5: If $m = \min(m,n,p)$, then the (m,n,p)-matrix multiplication problem has $(2,np/m^2)$-flow $\Omega(m^2)$ bits.

Proof: As in the proof of Lemma 4, there is no loss of generality in assuming $m \le c_1 n$ and $p \le c_1 m$ for a constant $0 < c_1 \le c/8 \le 1/64$. A $(2,I,np/m^2)$-partition is performed. There are exactly m^2 input bits from Y in each block. Moreover, there must exist a block with no more than $4m^3/p \le cm^2/2$ input bits from X and $4m^3/n \le cm^2/2$ output bits. Consider such a block. Partition the elements of X, Y and Z respectively into two sets, according to whether they are inside the block or not. The binary matrices $\mathfrak{x}$, $\mathfrak{y}$ and $\mathfrak{z}$ are defined as for Lemma 4. We shall show that regardless of the locations of the m^2 ones in $\mathfrak{y}$, $\Omega(m^2)$ bits have to cross of the boundary of the block. The proof is divided into two cases:

Case 1: There exists rows $Y(l_t,*)$, $1 \le t \le cm$, such that $\sum_{t=1}^{cm} \|\mathfrak{y}(l_t,*)\| \ge cm^2$: There must also exist rows $Z(k_t,*)$, $1 \le t \le cm$, such that $\sum_{t=1}^{cm} \|\mathfrak{z}(k_t,*)\| < c^2m^2/2$. Set X to the permutation matrix such that $Y(l_t,*)$ is mapped into $Z(k_t,*)$. Since there are no more than $c^2m^2/2$ output bits in the block, at least $(c-(c^2/2))m^2$

$= \Omega(m^2)$ bits must flow out of it in order that the outputs outside are computed correctly.

Case 2: The conditions of Case 1 is not satisfied: Let $\overset{\bullet}{\mathcal{Y}}$ be the largest submatrix of $\mathcal{Y}$ which has no all-zero rows or columns. By Lemma 3 there must exist a family of disjoint sets of columns of $\overset{\bullet}{\mathcal{Y}}$, $\{\overset{\bullet}{\mathcal{Y}}^{(t)} : 1 \le t \le s \le p\}$, such that

(a) $\|S_t\| = \| \bigvee_{\overset{\bullet}{\mathcal{Y}}(*,j) \in \overset{\bullet}{\mathcal{Y}}^{(t)}} \overset{\bullet}{\mathcal{Y}}(*,j)\| \ge cm$ and $\sum_{\overset{\bullet}{\mathcal{Y}}(*,j) \in \overset{\bullet}{\mathcal{Y}}^{(t)}} \|\overset{\bullet}{\mathcal{Y}}(*,j)\| \le 3\|S_t\|$ for all $1 \le t \le s$,

(b) $\sum_{t=1}^{s} \|S_t\| \ge cm^2$.

Let $J_t = \{j : \overset{\bullet}{\mathcal{Y}}(*,j) \in \overset{\bullet}{\mathcal{Y}}^{(t)}\}$. Consider the inner products $Z(t,j) = X(t,*) \cdot Y(*,j)$ for all $j \in J_t$. The quantity

$$n_t \triangleq \sum_{1 \le k \le n} 1(\exists j \in J_t : X(t,k) \oplus \mathcal{Y}(k,j) = 1)$$

is at least $|\|S_t\| - \|X(t,*)\||$. Next, we apply the following:

Fact [Pang85]: Given two binary sequences $\mathbf{x} = (x_1,\ldots,x_n)$ and $\mathbf{y} = (y_1,\ldots,y_n)$. Let the $2n$ bits be evenly distributed between two processors A and B. However, it is not necessary that all bits from one sequence are allocated to the same processor. Let $r \triangleq \sum_{i=1}^{n} 1(x_i$ and y_i are distributed to different processors). Then in order that A or B computes the inner product of $\mathbf{x}$ and $\mathbf{y}$ correctly for all $\mathbf{x} \in \{0,1\}^n$ and $\mathbf{y} \in \{0,1\}^n$, at least r bits must be exchanged between them. □

Hence, to compute $Z(t,j)$, $j \in J_t$, at least n_t bits must be exchanged via the boundary of the block. (Note that this is independent of whether the output is inside the block or not.) To compute the inner products $Z(t,j)$ for all $j \in J_t$ and $1 \le t \le s$, the total amount of communications is at least

$$\begin{aligned} \sum_{t=1}^{s} n_t &\ge \sum_{t=1}^{s} |(\|S_t\| - \|X(t,*)\|)| \\ &\ge |\sum_{t=1}^{s} (\|S_t\| - \|X(t,*)\|)| \\ &\ge (c - (c/2))m^2 = \Omega(m^2). \square \end{aligned}$$

A similar result can be proved with the roles of m and p interchanged. With these information flow bounds, the area-time lower bounds follow. The upper bound proofs in the following theorem are similar to that of [Lott84] and can also be found in [Pang85].

Theorem 2: Any circuit that performs (m,n,p)-matrix multiplication must satisfy $AT^2 = \Omega(mnpq)$ and $AT = \Omega(mnp)$, where $q = \min(m,n,p)$. Moreover, there are circuits that achieve:

(1) $p \geq n \geq m: A = O(pn\log n\log p)$ and $T = O(m+\log p)$.

(2) $p \geq m \geq n: A = O(pm\log m\log p)$ and $T = O(n+\log p)$.

(3) $n \geq p \geq m: A = O(pn\log n\log p)$ and $T = O(m+\log n)$. □

We now proceed to the area lower bounds. The proof of the following lemma is similar to that of Lemma 3.

Lemma 6: Suppose $cm \geq n$ for a constant $0 \leq c \leq 1/4$. Let Q be an $m\times m$ binary matrix such that $\|Q\| = m^2/2$, then regardless of the locations of the ones in Q, one of the following is satisfied:

(i) There exists a disjoint collection of n groups of rows of Q, $\{Q^{(i)} : 1 \leq i \leq n\}$, such that $\sum_{i=1}^{n} \bigvee_{Q(r,*)\in Q^{(i)}} Q(r,*) \geq (1 + c_1)mn/2$, where $0 < c_1 \leq c$.

(ii) There exists a disjoint collection of n groups of columns of Q, $\{Q^{(i)} : 1 \leq i \leq n\}$, such that $\sum_{i=1}^{n} \bigvee_{Q(*,r)\in Q^{(i)}} Q(*,r) \geq (1 + c_1)mn/2$. □

The following lemma is simply a restatement of the bisection information flow bound in [Sava81].

Lemma 7: If $n \geq m$ and $p \geq m$, then the (m,n,p)-matrix multiplication problem has bisection information flow $\Omega(\min(n,p)\cdot m)$ bits. □

A similar result can be proved with the roles of m and p interchanged. The next lemma presents a bisection information flow result that also gives the AT^2 lower bound in [Lott84].

Lemma 8: If $p \geq n$ and $m \geq n$, then the (m,n,p)-matrix multiplication problem has bisection information flow $\Omega(\min(m,p)\cdot n)$ bits.

Proof (Sketch): Since the bound to be proved is independent of $\max(m,n,p)$, there is no loss of generality in assuming that $p = m$. Moreover, we can also assume that there exists a constant $0 < c \leq 1/4$ such that $cm \geq n$.

Bisect the given circuit across time, so that there are exactly $m^2/2$ outputs in each block. Let B_1 and B_2 be the two blocks, where the former precedes the latter in time. One of them must have $mn/2$ or more inputs from Y. We first consider the case where this block is B_1. Construct the binary matrix whose (i,j)th position is 1 if $Y(i,j)$ is read inside B_2, and 0 otherwise. Lemma 6 guarantees that X can be set to a permutation matrix such that the total number of pairs (i,j) such that $Y(i,j)$ is read inside B_2, and that $Z(i,j)$ is mapped from a distinct element from Y is more than $(1+c_1)mn/2$. Since no more than $mn/2$ of these are inside the block, at least $c_1mn/2$ bits have to flow in.

If B_2 is the block with $mn/2$ or more inputs from Y, it can be similarly shown that at least $c_1mn/2$ outputs are required before they are even input, an obvious contradiction. □

With the information flow bounds of Lemmas 7 and 8, the area bound follows readily. This lower bound is achieved by running the algorithm of Theorem 2 on the mesh-of-processors instead of the mesh-of-trees.

Theorem 3: Any circuit that computes (m,n,p)-matrix multiplication must satisfy $A = \Omega(mnp/r)$, where $r = \max(m,n,p)$. Moreover, this bound is achievable. □

Appendix A Packing Lemma for Binary Matrices

Before proving the packing lemma (Lemma 3), we need to establish the following result:

Lemma A1: Let Q be a $k \times l$ binary matrix, where $k \ge n$, $l \ge n$ and $0 < c \le 1/4$. If $\|Q\| \ge 4cn^2$, $\|Q(i,*)\| > 0$ for all i and $\|Q(*,j)\| > 0$ for all j, then one of the following must be satisfied:

(1) There exist cn rows $Q(i_t,*)$ such that

$$\sum_{t=1}^{cn} \|Q(i_t,*)\| \ge cn^2. \tag{A1}$$

(2) There exists a set of s columns $Q^{(s)} = \{Q(*,j_t) : 1 \le t \le s\}$, where $1 \le s \le n$, such that

$$\|\bigvee_{t=1}^{s} Q(*,j_t)\| \ge cn \quad \text{and} \tag{A2}$$

$$\sum_{t=1}^{s} \|Q(*,j_t)\| \le 3\, \|\bigvee_{t=1}^{s} Q(*,j_t)\|, \tag{A3}$$

where $\vee$ is the component-wise logic OR operation.

Proof: Suppose condition (1) is not satisfied, we shall show that the set of columns satisfying condition (2) can be constructed. Let j_1 be such that $\|Q(*,j_1)\| = \max_{1 \le j \le l} \|Q(*,j)\|$. Let $\|Q(*,j_1)\| = n_1$ and $Q(i_t,j_1) = 1$ for $1 \le t \le n_1$. If $n_1 \ge cn$, then the set $Q^{(1)} = \{Q(*,j_1)\}$ satisfies (A2) and (A3) and the lemma is proved. If not, consider all the remaining columns. If for all $j \ne j_1$, $\sum_{t=1}^{n_1} \|Q(i_t,j)\| > \|Q(*,j)\|/2$, then

$$\sum_{j \ne j_1} \sum_{t=1}^{n_1} \|Q(i_t,j)\| > \Big(\sum_{j \ne j_1} \|Q(*,j)\|\Big)/2$$
$$> (4cn^2 - cn)/2 > cn^2.$$

In other words, there is a collection of less than cn rows which totally contains more than cn^2 ones – – – a contradiction to the fact that Q does not satisfy (A1). Hence there must exist $1 \le j_2 \le l$ such that

$$\|Q(*,j_1) \vee Q(*,j_2)\| - \|Q(*,j_1)\| \ge \|Q(*,j_2)\|/2. \tag{A4}$$

Denote $Q_2 = Q(*,j_1) \vee Q(*,j_2)$, the condition $\|Q_2\| \ge (\|Q(*,j_1)\| + \|Q(*,j_2)\|)/3$ is satisfied. If $\|Q_2\| \ge cn$, then the lemma is proved because the set of columns $Q^{(2)} = \{Q(*,j_1), Q(*,j_2)\}$ satisfies (A2) and (A3). If not, then $2cn > \|Q(*,j_1)\| + \|Q(*,j_2)\|$ and the above procedure is repeated on the remaining columns. In general, let the set of columns $Q^{(q)} = \{Q(*,j_1), ..., Q(*,j_q)\}$ be constructed and (A2) is not yet satisfied. Define the k-vector $Q_q = \bigvee_{t=1}^{q} Q(*,l_t)$ and let $||Q_q|| = n_q$. Let the set of nonzero bit positions of Q_p be $I_q = \{i_1,...,i_{n_q}\}$. Then

$$cn > ||Q_q|| \geq (\sum_{t=1}^{q} ||Q(*,j_t)||)/3. \tag{A5}$$

Hence

$$\sum_{Q(*,j) \in Q^{(q)}} ||Q(*,j)|| > 4cn^2 - 3cn$$

$$= 3cn^2 \quad \text{for large enough } n.$$

If for all $Q(*,j) \in Q^{(q)}$, $\sum_{i \in I_q} ||Q(i,j)|| > ||Q(*,j)||/2$, then

$$\sum_{Q(*,j) \in Q^{(q)}} \sum_{i \in I_q} ||Q(i,j)|| > (\sum_{Q(*,j) \in Q^{(q)}} ||Q(*,j)||)/2 > cn^2.$$

In other words, there is a collection of less than cn rows which totally contains more than cn^2 ones --- a contradiction. Therefore, there must exist $1 \leq j_{q+1} \leq l$ such that

$$||Q_q \vee Q(*,j_{q+1})|| - ||Q_q|| \geq ||Q(*,j_{q+1})||/2. \tag{A6}$$

Let $Q_{q+1} = Q_q \vee Q(*,j_{q+1})$ and $Q^{(q+1)} = Q^{(q)} \cup \{Q(*,j_{q+1})\}$. If $||Q_{q+1}|| \geq cn$, then since $||Q(*,j_{q+1})|| \leq cn$, $Q^{(q+1)}$ satisfies (A2) and (A3) and the lemma is proved. If not, (A5) and (A6) imply that

$$cn > ||Q_{q+1}|| \geq (\sum_{t=1}^{q+1} ||Q(*,j_t)||)/3$$

and the construction can be continued on the remaining columns. Since $||Q|| \geq 4cn^2$ and no more than $3cn$ of them are used before the construction is complete, the process must terminate and Lemma A1 is proved. □

With this result, the packing lemma can now be proved.

Lemma 3: Let Q be an $k \times l$ binary matrix, where $k \geq n$, $l \geq n$ and $0 < c \leq 1/8$. If $||Q|| = n^2$, $||Q(i,*)|| > 0$ for all i and $||Q(*,j)|| > 0$ for all j, then one of the following must be satisfied:

(1) There exist cn rows $Q(i_t,*)$ such that $\sum_{t=1}^{cn} ||Q(i_t,*)|| \geq cn^2$.

(2) There exists a family of disjoint sets of columns of Q, $\{Q^{(t)} : 1 \leq t \leq s\}$, where $s \leq n$. Define $S_t = \bigvee_{Q(*,j) \in Q^{(t)}} Q(*,j)$, then for each $1 \leq t \leq s$, the following are satisfied:

(a) $||S_t|| \geq cn$ and $\sum_{Q(*,j) \in Q^{(t)}} ||Q(*,j)|| \leq 3||S_t||$ for $1 \leq t \leq s$.

(b) $\sum_{t=1}^{s} ||S_t|| \geq cn^2$.

Proof: Suppose condition (1) is not satisfied, we shall show, by induction using Lemma A1, that the family of sets of columns satisfying condition (2) can be constructed. Let $Q_0 = Q$. Clearly, Q_0 satisfies the conditions of Lemma A1. Hence $Q^{(1)}$ can be constructed. Let $Q_1 = Q_0 - Q^{(1)}$, where "−" is the set difference operation, and Q_0 is considered as a set of k-vectors which are its columns. If $||S_1|| \geq cn^2$, then the lemma is proved. If not, then $||Q_1|| \geq (1 - 3c)n^2$. Since $c \leq 1/8$, $1 - 3c \geq 4c$. Hence $||Q_1|| \geq 4cn^2$, i.e. it satisfies the conditions of Lemma A1. $Q^{(2)}$ can therefore be constructed. If $||S_1|| + ||S_2|| \geq cn^2$, then the lemma is proved. If not, then define $Q_2 = Q_1 - Q^{(2)}$ and $||Q_2|| \geq (1 - 3c)n^2 \geq 4cn^2$ follows. In general, suppose after the construction of $Q^{(1)}, ..., Q^{(i)}$, it is still the case that $\sum_{t=1}^{i} ||S_t|| < cn^2$, then by Lemma A1, it must be true that

$$\sum_{t=1}^{i} \sum_{Q(*,r) \in Q^{(t)}} ||Q(*,r)|| < 3cn^2.$$

The next set of columns $Q^{(i+1)}$ can then be constructed. Since $||Q|| = n^2$ and no more than $3cn^2$ are required for the construction, the process must terminate and Lemma 3 is proved. □

References

[Angl82] D. Angluin, "VLSI: On the Merits of Batching," Manuscript 1982.

[BP86] G. Bilardi and F.P. Preparata, "Area-Time Lower Bound Techniques with Application to Sorting," to appear in *Algorithmica*, Jan. 1986.

[EGP84] A. El Gamal, J.W. Greene and K.F. Pang, "VLSI Complexity of Coding," *Proc MIT Conf. Advanced Research in VLSI*, Jan. 1984; also to appear in *IEEE Trans. Information Theory*.

[Grig76] D. Grigoryev, "An Application of Separability and Independence notions for proving Lower Bounds on circuit complexity," *Notes of Scientific Seminars, Steklov Math. Inst.*, vol.60, 1976, pp.35-48. (In Russian. Translation of Section 2 obtained from J. Savage.)

[Lott84] G. Lotti, "Area-Time Tradeoff for Rectangular Matrix Multiplication in VLSI Models," *Information Processing Letters*, vol.19, Aug. 1984, pp.95-98.

[Pang85] K.F. Pang, "Complexity Results in VLSI and Distributed Computing," PhD Thesis, EE Dept., Stanford University, June 1985.

[Sava81] J.E. Savage, "Area-Time Tradeoffs for Matrix Multiplication and related problems in VLSI models," *J. Computer and System Sci.*, vol.22, 1981, pp.230-242.

[Thom80] C.D. Thompson, "A Complexity Theory for VLSI," PhD Thesis, CS Department, Carnegie-Mellon University, August 1980; also appeared as Technical Report CMU-CS-80-140.

[Ullm83] J.D. Ullman, *Computational Aspects of VLSI*, Computer Science Press, 1983.

A MATRIX FACTORIZATION APPROACH TO THE PARALLEL SOLUTION OF FIRST-ORDER LINEAR RECURRENCES

GERARD G. L. MEYER and LOUIS J. PODRAZIK
Electrical Engineering and Computer Science Department
The Johns Hopkins University
Baltimore, Maryland 21218

ABSTRACT

In this paper, we present a parametrized procedure for the parallel solution of first-order linear recurrences. We analyze the parallelism exhibited by the procedure as a function of its parameters, and we show how to determine the parameters as a function of the number of processing elements in the computational network to produce minimal computational time.

1. INTRODUCTION

Many algorithms have been proposed to solve linear recurrences in parallel, e.g., [CHE75], [CHE78], [KUC76], [HYA79], [GAJ81], [KOG73], [CAR84], [LAK85]. The efficiency of each of these techniques is highly dependent upon the number p of processors to be used. In particular, no single approach is most efficient for all p in the range $1 \lesssim p \lesssim n$, where n is the size of the recurrence. Unlike such approaches, our technique parametrizes the algorithm. By choosing the optimal parameters as a function of the number of processors, we are able to design an algorithm which is both time and processor efficient. This is accomplished by optimizing the tradeoff between minimizing the total number of computations and maximizing concurrency through proper choice of algorithm parameters. Thus, by a parametrized approach, we are able to recover existing results and in addition, extend upon them.

Due to space limitations, all proofs have been deleted, and only first order linear recurrences whose size n is a power of two, that is, $n = 2^\alpha$ are considered.

The organization of this work is as follows : in Section 2, the first-order linear recurrence problem is stated in matrix notation. Section 3 presents the parametrized algorithm with the optimal parameters being determined in Section 4. Finally, in Section 5 conclusions are presented.

2. PRELIMINARIES

The problem considered in this paper is the well-known first order linear recurrence problem.

Problem 1: Given $n = 2^\alpha$, given b_i for $i = 1,2,...,n$, and given a_i for $i = 2,3,...,n$, compute $x_1, x_2, \ldots, x_n$ such that $x_1 = b_1$ and

$x_i = a_i x_{i-1} + b_i$ for $i = 2,3,\dots,n$.

Problem 1 may be rewritten using matrix notation.

Problem 2: Given $b^t = (b_1,b_2,\dots,b_n)^t$, and the $n \times n$ lower triangular matrix A that consists of n^2 scalars a_{ij} defined for all i and j in $[1,2,\dots,n]$ by

$$a_{ij} = \begin{cases} 1 & \text{if } i = j \\ -a_i & \text{if } i = j+1 \\ 0 & \text{otherwise} \end{cases}$$

compute $x^t = (x_1,x_2,\dots,x_n)^t$ such that $Ax = b$.

We now present a parametrized factorization of A which produces an efficient parallel solution to Problem 2.

3. PARAMETRIZED ALGORITHM

Let W be the family of all integer sequences $\{w_i\} = \{w_1,w_2,\dots,w_v\}$ that satisfy

$$\sum_{i=1}^{v} w_i = \alpha \text{ and } w_i \geq 1,\ i = 1, 2, \dots, v.$$

Given a sequence $\{w_i\}$ in W, let the integers λ_i, i in $[1,2,\dots,v]$ be defined as

$$\lambda_i = \begin{cases} 0 & \text{if } i = 1 \\ \sum_{m=1}^{i-1} w_m & \text{if } i > 1 \end{cases} \tag{3.1}$$

and for i in $[1,2,\dots,v]$ and j in $[1,2,\dots,2^{w_i}-1]$, let the index sets $s_{i,j}$ be defined as

$$s_{i,j} = \{ j2^{\lambda_i} + q2^{\lambda_i+w_i} : q = 0,1,\dots,\frac{n}{2^{\lambda_i+w_i}}-1 \}. \tag{3.2}$$

For k in $[1,2,\dots,n-1]$, let e_k be the k-th column of the $n \times n$ identity matrix I. Now, for $k \in s_{i,j}$, i in $[1,2,\dots,v]$ and j in $[1,2,\dots,2^{w_i}-1]$, and λ_i, i in $[1,2,\dots,v]$, let h_{k,λ_i} be the vector in E^n defined as

$$h_{k,\lambda_i} = \sum_{q=1}^{2^{\lambda_i}} \left[\prod_{m=k+1}^{k+q} a_m \right] e_{k+q} \tag{3.3}$$

Finally, for i in $[1,2,\ldots,v]$ and j in $[1,2,\ldots,2^{w_i}-1]$, let $Q_{i,j}$ be the nxn strictly lower triangular matrix defined by

$$Q_{i,j} = \sum_{k \varepsilon s_{i,j}} h_{k,\lambda_i} e_k^t \tag{3.4}$$

The following lemma shows that to each sequence $\{w_i\}$ in W corresponds a factorization of A.

Lemma 1 : If $\{w_i\}$ is in W, then

$$A = \prod_{i=1}^{v} \prod_{j=1}^{2^{w_i}-1} (I - Q_{i,j}). \tag{3.5}$$

Note that if $v = 1$, that is, $\{w_i\} = \{w_1\} = \{\alpha\}$ then $\lambda_1 = 0$, the index sets $s_{i,j}$ are defined for $i = 1$ and $j = 1,2,\ldots,n-1$, and $s_{1,j} = \{j\}$. It follows that for j in $[1,2,\ldots,n-1]$, $h_{j,0} = h_j$, where h_j is the j-th column of I-A. Thus, when $v = 1$, we define n-1 matrices $Q_{1,j}$, $j = 1,2,\ldots,n-1$ where each matrix $Q_{1,j}$ is zero except for the j-th column that is equal to the j-th column of the matrix I-A. Thus, when $v = 1$, Eq. (3.5) reduces to

$$A = \prod_{j=1}^{n-1} (I - Q_{1,j}) = \prod_{j=1}^{n-1} (I - h_j e_j^t),$$

that is, the Householder factorization of triangular matrices as a product of elementary triangular matrices [HOU65].

We now present those properties of the matrices $Q_{i,j}$ which allow us to develop an efficient parallel approach to solve Problem 2.

Lemma 2 : Let $\{w_i\}$ be in W and let $\{Q_{i,j}\}$ be the corresponding sequence of matrices, then (i) $Q_{i,j}$ has $n/2^{w_i}$ non-zero rows, each containing exactly one non-zero element and (ii) $(I - Q_{i,j})^{-1} = I + Q_{i,j}$.

Lemma 2 implies

$$A^{-1} = \prod_{i=v}^{1} \prod_{j=2^{w_i}-1}^{1} (I + Q_{i,j}), \tag{3.6}$$

and consequently, the following algorithm solves Problem 2.

<u>Algorithm</u> <u>1</u>: Given b, a sequence of integers $\{w_i\}$ in W and $Q_{i,j}$ for i in $[1,2,\ldots,v]$, j in $[1,2,\ldots,2^{w_i}-1]$.

```
Step 0:  Let $y = b$.
Step 1:  for $i = 1$ until $v$ DO
Step 2:          for $j = 1$ until $(2^{w_i} - 1)$ DO
Step 3:                  $x = y + Q_{i,j}y$
Step 4:                  $y = x$
Step 5:          next j
Step 6:  next i
```

Step 3 of Algorithm 1 consists of the matrix-vector product $Q_{i,j}y$ followed by the vector addition $x = y + Q_{i,j}y$. We know from Lemma 2 that the matrix-vector product requires $n/2^{w_i}$ multiplications, each of which can be performed in parallel, and that the vector addition requires $n/2^{w_i}$ additions which can also be performed in parallel. Each computational operation of Step 3 consists of either a parallel addition or multiplication. Thus, Algorithm 1 requires all processors to perform the same operation at the same time, resulting in an SIMD (Single-Instruction, Multiple-Data Stream) type of computation.

To each $\{w_i\}$ in W corresponds a specific implementation of Algorithm 1 and associated computational time. Given n, p and $\{w_i\}$, let $T(n,p,\{w_i\})$ be the computational time required by Algorithm 1 to compute the solution to the recurrence system.

<u>Lemma</u> <u>3</u> : Let $n = 2^{\alpha}$, and $\{w_i\}$ be in W, then

$$T(n,p,\{w_i\}) = \sum_{i=1}^{v} 2(2^{w_i}-1) \left\lceil \frac{n}{2^{w_i}p} \right\rceil . \tag{3.7}$$

Obviously, the best algorithm for a given n and p corresponds to a choice of $\{w_i\}$ that minimizes $T(n,p,\{w_i\})$. Thus, the optimal parameters $\{\tilde{w}_i\}$ are obtained by solving the integer programming problem below.

Problem 3 : Given $n = 2^{\alpha}$ and p, find $\{\tilde{w}_i\}$ in W such that $T(n,p,\{\tilde{w}_i\}) \leq T(n,p,\{w_i\})$ for all $\{w_i\}$ in W.

Note that given n and p, the optimal choice $\{\tilde{w}_i\}$ is based on a trade-off between the number $\sum_{i=1}^{v} 2^{w_i}-1$ of matrix-vector products $(I + Q_{i,j})y$ and the number $2\left\lceil \frac{n}{2^{w_i}p} \right\rceil$ of steps needed to obtain each matrix product using p processors.

4. OPTIMAL PARAMETER DETERMINATION

Although Problem 3 may be solved using any suitable integer programming algorithm, it is not unrealistic to use a direct search. However, rather than calculating the time associated with each of the $2^{\alpha-1}$ $\{w_i\}$ in W and determining the optimal, we make the following observation: $T(n,p,\{w_i^1\}) = T(n,p,\{w_i^2\})$ whenever $\{w_i^1\}$ and $\{w_i^2\}$ are in W and the two sequences are a permutation of each other. Thus, we do not have to search all of W, rather we need only consider sequences in W whose elements are monotone nondecreasing. These sequences are the unrestricted partitions of α and are easily found using a procedure given in [BEC64]. The number of unrestricted partitions of α, $N(\alpha)$, is bounded from above by $\frac{1}{4\alpha(3)^{0.5}} \exp\left[\pi\left[\frac{2\alpha}{3}\right]^{1/2}\right]$ and therefore small. For example, when $n = 2^{16}$, $N(16) \leq 258$ and when $n = 2^{16}$, $N(32) \leq 9{,}037$.

In order to obtain closed form expressions for the optimal sequence $\{\tilde{w}_i\}$, and the corresponding minimum computational time $T(n,p,\{\tilde{w}_i\})$, we make one additional assumption : the number of processors is a power of two, i.e., $p = 2^{\beta}$. In this case, $n/(2^{w_i}p) = 2^{\alpha-w_i-\beta}$. If $\beta \leq \alpha-w_i$, then $n/(2^{w_i}p)$ is an integer, otherwise it is less than one. Thus, we have $\left\lceil \frac{n}{2^{w_i}p} \right\rceil = \mathrm{Max}\left[1, \frac{n}{2^{w_i}p}\right]$. It then follows from Eq. (3.7) that

$$T(n,p,\{w_i\}) = \sum_{i=1}^{v} 2(2^{w_i}-1)\ \mathrm{Max}\left[1, \frac{n}{2^{w_i}p}\right]. \tag{4.1}$$

Thus, when both n and p are powers of 2, the optimal parameters $\{\tilde{w}_i\}$ are obtained by solving the integer programming problem below.

Problem 4 : Given $n = 2^{\alpha}$ and $p = 2^{\beta}$, find $\{\tilde{w}_i\}$ in W such that $T(n,p,\{\tilde{w}_i\}) \leq T(n,p,\{w_i\})$ for all $\{w_i\}$ in W.

The following two lemmas give the solution of Problem 4 as a function of the relative values of n and p.

Lemma 4 : Let $n = 2^{\alpha}$ and $p = 2^{\beta}$ be such that $1 \leq p \leq \frac{n}{2}$. A sequence of integers $\{\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_{\tilde{v}}\}$ solves Problem 4 if and only if (i) $\tilde{v} = \left\lceil \frac{\alpha}{\alpha-\beta} \right\rceil$, (ii) if $\tilde{v} = \frac{\alpha}{\alpha-\beta}$, then $\tilde{w}_i = \alpha-\beta$ for all i in $[1,2,\ldots,\tilde{v}]$ and (iii) if $\tilde{v} \neq \frac{\alpha}{\alpha-\beta}$, then $\{\tilde{w}_i\}$ consists of $\tilde{v}-1$ $\tilde{w}_i$'s equal to $\alpha-\beta$ and one residual $\tilde{w}_i$ equal to $\alpha - (\tilde{v}-1)(\alpha-\beta)$.

Lemma 5 : Let $n = 2^{\alpha}$ and $p = 2^{\beta}$ be such that $p \geq \frac{n}{2}$. A sequence of integers $\{\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_{\tilde{v}}\}$ solves Problem 4 if and only if $\tilde{v} = \alpha$ and $\tilde{w}_i = 1$ for all i in $[1,2,\ldots,\tilde{v}]$.

Note that if $p = 1$, then $\tilde{v} = 1$ and $\tilde{w}_1 = \alpha$; if $1 < p \leq n^{1/2}$, then $\tilde{v} = 2$, and Problem 4 has two solutions, $\tilde{w}_1 = \alpha-\beta$, $\tilde{w}_2 = \beta$ and $\tilde{w}_1 = \beta$, $\tilde{w}_2 = \alpha-\beta$; if $n^{1/2} < p < \frac{n}{2}$, then $3 \leq \tilde{v} \leq \alpha-1$, with $\{\tilde{w}_i\}$ satisfying Lemma 4; finally, if $p \geq \frac{n}{2}$, then $\tilde{v} = \alpha$ and $\tilde{w}_i = 1$ for all i in $[1,2,\ldots,\alpha]$.

We can now give closed form expressions for $T(n,p,\{\tilde{w}_i\})$ as a function of the relative values of n and p.

Theorem 1 : Given $n = 2^{\alpha}$ and $p = 2^{\beta}$, let $\gamma = \left\lfloor \frac{\alpha}{\alpha-\beta} \right\rfloor$. Then,

$$T(n,p,\{\tilde{w}_i\}) = \begin{cases} 2n-2 & \text{if } p = 1 \\ \frac{4n}{p} - \frac{2n}{p^2} - 2 & \text{if } 1 < p \leq n^{1/2} \\ 2(\gamma+1)\frac{n}{p} - \frac{2n^{\gamma}}{p^{\gamma+1}} - 2\gamma & \text{if } n^{1/2} < p < \frac{n}{2} \\ 2\log_2 n & \text{if } p \geq \frac{n}{2} \end{cases}$$

In order to examine the tradeoff performed in the determination of the optimal sequence $\{\tilde{w}_i\}$, we now present an example for the case $n = 8$, $p = 2$.

Example 1 : Let $n = 8$ and let $p = 2$. In that case, $\tilde{v} = 2$ and the sequences $\{1,2\}$ and $\{2,1\}$ are optimal, with $T(8,2,\{1,2\}) = T(8,2,\{2,1\}) = 10$. When either optimal sequence is used, Algorithm 1 computes the solution by performing one matrix-vector product requiring 4 time steps and 3 additional matrix-vector products, each requiring 2 time steps. When $v = 1$, the sequence $\{3\}$ produces the time $T(8,2,\{3\}) = 14$, with Algorithm 1 performing 7 matrix-vector products, each requiring 2 time steps. When v = 3, the sequence {1,1,1} produces the time T(8,2,{1,1,1}) = 12, with Algorithm 1 performing only 3 matrix-vector products, each requiring 4 time steps.

A special case of Problem 1 occurs when $a_i = 1$, for all i in [1,2,...,n]. In that case, Problem 1 is closely related to the prefix problem and reduces to computing the cascade sums (b_1+b_2), $(b_1+b_2+b_3)$, ..., $(b_1+b_2+ \cdots +b_n)$ in parallel using p processors. The following corollary is a direct consequence of Theorem 1.

Corollary 1 : Let $n = 2^{\alpha}$, $p = 2^{\beta}$, $a_i = 1$, for all i in [1,2,...,n] and let $\gamma = \left\lfloor \frac{\alpha}{\alpha-\beta} \right\rfloor$, then

$$T(n,p,\{\tilde{w}_i\}) = \begin{cases} n-1 & \text{if } p = 1 \\ \dfrac{2n}{p} - \dfrac{n}{p^2} - 1 & \text{if } 1 < p \leq n^{1/2} \\ (\gamma+1)\dfrac{n}{p} - \dfrac{n^{\gamma}}{p^{\gamma+1}} - \gamma & \text{if } n^{1/2} < p < \dfrac{n}{2} \\ \log_2 n & \text{if } p \geq \dfrac{n}{2} \end{cases}$$

One may compare the results of Corollary 1 with the result $T = \frac{n}{2p}(2 + \log_2 p)$ presented in [LAK85].

5. CONCLUSIONS

A parallel algorithm to solve first order linear recurrences was developed for use on p processors, where $1 \leq p \leq n$. The algorithm is an SIMD (single-instruction, multiple-data stream) type of computation in that at each step of the calculation all active processors are performing the same operation.

By using a parametrized approach to solve the problem, we have been able to find the minimal computational time associated with a particular class of factorizations of A. Within this class we are able to recover existing results. In particular, when $p < n^{1/2}$, our time is equivalent to that presented by Gajski [GAJ81] for an SIMD machine. Furthermore, when p = n/2, the results of Kogge and Stone [KOG73] and those of Chen and Kuck [CHE75], [KUC76] are retrieved. However, for $n^{1/2} < p < n/2$, the time required by Algorithm 1 is less than any other approach known to lie within this class of factorizations.

Finally, a generalization of the technique presented here leads to similar results for m-th order linear recurrence systems.

6. AKNOWLEDGEMENT

This work was supported by the Air Force Office of Scientific Research under Contract AFOSR-85-0097.

7. REFERENCES

[BEC64] Beckenbach, E.F., Applied Combinatorial Mathematics, John Wiley and Sons, Inc., New York, London and Sydney, 1964.

[CAR84] Carlson, D.A., and Sugla, B., Time and Processor Efficient Parallel Algorithms for Recurrence Equations and Related Problems Proceedings of the 1984 International Conference on Parallel Processing, August 21-24, 1984, pp. 310-314.

[CHE75] Chen, S.C. and Kuck, D.J., Time and Parallel Processor Bounds for Linear Recurrence Systems, IEEE Trans. Computers., Vol. C-24, No.7, July 1975, pp. 701-717.

[CHE78] Chen, S.C., Kuck, D.J. and Sameh, A.H., Practical Parallel Band Triangular System Solvers, ACM Trans. on Mathematical Software, Vol. 4, No. 3, September 1978, pp. 270-277.

[GAJ81] Gajski, D. J., An Algorithm for Solving Linear Recurrence Systems on Parallel and Pipelined Machines, IEEE Trans. Computers., Vol. C-30, No. 3, March 1981, pp. 190-206.

[HOU65] Householder, A.S., The Theory of Matrices in Numerical Analysis, Blaisdell Publishing Company, New York, Toronto and London, 1965.

[HYA79] Hyafil, L. and Kung, H. T., Parallel Algorithms for Solving Triangular Linear Systems with Small Parallelism, Department of Computer Science Report, December 1974, Carnegie-Mellon University, Pittsburgh, PA 15212.

[KOG73] Kogge, P.M., and Stone, H.S., A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations IEEE Trans. Computers., Vol. C-22, No. 8, August 1973, pp. 786-793.

[KUC76] Kuck, D. J., Parallel Processing of Ordinary Programs, Advances in Computers., Vol. 15, Academic Press, New York, 1976, pp. 119-179.

[LAK85] Lakshmivarahan, S., and Dhall, S.K., New Parallel Algorithms for Solving First-Order and Certain Classes of Second-Order Linear Recurrences, Proceedings of the 1985 International Conference on Parallel Processing, August 20-23, 1985, pp. 843-845.

On Using Multiple Inverted Trees for Parallel Updating of Graph Properties [+]

Shaunak Pawagi

I. V. Ramakrishnan [++]

Department of Computer Science
University of Maryland
College Park, Maryland 20742

Abstract -- Fast parallel algorithms are presented for updating the distance matrix, shortest paths for all pairs and biconnected components for an undirected graph and the topological ordering of vertices of a directed acyclic graph when an incremental change has been made to the graph. The kinds of changes that are considered here include insertion of a vertex or insertion and deletion of an edge or a change in the weight of an edge. The machine model used is a parallel random access machine which allows simultaneous reads but prohibits simultaneous writes into the same memory location. The algorithms described in this paper require O(log n) time and use $O(n^3)$ processors. These algorithms are efficient when compared to previously known $O(\log^2 n)$ time start-over algorithms for initial computation of the above mentioned properties of graphs. The previous solution is maintained in multiple inverted trees (a rooted tree where a child node points towards its father) after a minor change the new solution is rapidly recomputed from these trees.

1. Introduction

Incremental graph algorithms deal with recomputing properties of graph after an incremental change has been made to the graph. Such recomputations are also referred to as "updating" graph properties. Sequential incremental algorithms for updating minimum spanning trees [12] connected components [3] and shortest paths [4] have appeared in the past. However, except in [7, 8], incremental algorithms have not been studied for any model of parallel computation. Parallel algorithms for several graph problems have been devised for an unbounded model of parallel random access machine (PRAM) (see [5, 6, 10, 11, 13]. In this model of computation all processors have access to a global memory and processors can simultaneously read from the same location but no two processors can simultaneously write into the same location. The algorithms developed on this model of computation provide us with a basis to compare the complexity of our incremental algorithms. In this paper we describe incremental algorithms for updating the distance matrix, shortest paths for all pairs and biconnected components of an undirected graph and topological ordering of the vertices of a directed acyclic graph (DAG). Our algorithms for updating these properties require O(log n)[†] time on a PRAM and therefore are efficient when compared to the start-over algorithms for initial computation of these properties that require $O(\log^2 n)$ time.

The kinds of minor modifications that are considered here are as follows. First, a vertex may be added along with the edges incident on it. Second, an individual edge may be deleted or added. If an edge has a weight associated with it then an increase or decrease in its weight is permitted. Note that an increase or decrease in the weight of an edge encompasses an edge deletion or insertion operation, because we can treat the nonexistent edges as having infinite weights.

[+] The support of the first author by the Air Force Office of Scientific Research under Contract F-49620-85-K-0009, and of the second author by the Office of Naval Research under Contract N00014-84-K-0530, and by the National Science Foundation under grant ECS-84-04399, is gratefully acknowledged.

[++] Present address: Dept. of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794.

[†] Throughout this paper, we use log n to denote $\lceil \log_2 n \rceil$.

An important aspect of incremental algorithms is the design of data structures to store the previous solution as well as some auxiliary information that is generated during the initial computation Such data structures should provide rapid access to the necessary information for efficient updates of the solution. It was shown in [8] that an inverted spanning tree can be used to update a minimum spanning tree, connected components and bridges of an undirected graph in O(log n) time on a PRAM. In this paper we use n such inverted spanning trees for updating the distance matrix, shortest paths for all pairs, biconnected components and topological ordering of the vertices. Specifically, in the distance matrix the i^{th} row defines a distance tree (DT) for the underlying graph and such a tree is rooted at vertex i. (A distance tree is in fact a shortest path tree for an unweighted graph.) Now, for a graph with n vertices, if we store n such trees (each having a different root) then we can quickly identify the pairs of vertices whose distances must be recomputed after the graph undergoes a minor change. In the case of shortest paths for all pairs, we store the shortest path trees as inverted trees. The longest path trees that are involved in the computation of topological ordering of the vertices of a DAG, are stored as inverted trees. In order to update biconnected components efficiently we store n inverted spanning trees, one for each graph G_i that is obtained from the original graph G by deleting vertex i.

The rest of the paper is organized as follows. In Section 2 we describe some graph-theoretic preliminaries adopting the framework in [13]. In Section 3 we describe our algorithms for updating the distance matrix. In Sections 4 we extend the ideas of Section 3 to update shortest paths for all pairs. Concluding remarks appear in section 5, where we discuss updating a topological ordering of vertices of a DAG and biconnected components.

2. Preliminaries

In order to describe our algorithms to update graph properties we now present some graph theoretic preliminaries.

Let G=(V,E) denote a *graph* where V is a finite set of vertices and E is a set of pairs of vertices called edges. Throughout this paper we assume that $V=\{1,2,...,n\}$, $|V|=n$ and $|E|=m$. We denote the undirected edge from a to b by (a,b) and the directed edge joining them by $<a,b>$. A *path* in G joining two vertices i_0 and i_k is defined as a sequence of vertices $(i_0,i_1,i_2, \ldots, i_k)$ such that all of them are distinct and for each p, $0 \leq p < k$, (i_p,i_{p+1}) is an edge of G. If $i_0 = i_k$ then the path is called a cycle. We denote an undirected path from vertex u to vertex v by [u-v] and a directed path by $[u \rightarrow v]$. We assume the standard definitions for connectivity, component, tree, adjacency matrix and a spanning tree of a graph. A *rooted* tree T has a distinguished vertex called *root*. We say that vertex u is an *ancestor* of vertex v if u is on the path from v to the root of the tree. A *descendant* of a vertex is defined similarly. The *father* of a vertex is its immediate ancestor. The *depth* of a vertex is its distance from the root. The *lowest common ancestor* (LCA) of vertices x and y in T is the vertex z such that z is a common ancestor of x and y, and any other common ancestor of x and y in T is also an ancestor of z in T. If the edges of a tree T are oriented from a vertex to its father then T is an *inverted* tree. An *inverted spanning tree* (IST) and an *inverted spanning forest* (ISF) are defined similarly. An IST is completely specified by its root and the father for each vertex. Let the function F denote the father for each vertex. Let $F(r) = r$, where r is the root.

As we will see later on, our update algorithms require the following computations on an inverted tree; (i) for every vertex, its path to the root, (ii) identification of the two different subtrees that are created by deleting an edge from the tree, (iii) determining the ancestors and descendants for every vertex. Tsin and Chin [13] have described an efficient technique for computing functions associated with an inverted tree. The following lemma states their result.

Lemma 2.1: For an inverted tree having n vertices, the above mentioned functions can be computed in O(log n) time using $O(n^2)$ processors.

We now proceed to describe our incremental algorithms for updating the distance matrix.

3. Distance Matrix

In this section we consider updating the distance matrix of an undirected graph whose edges have uniform weights. For a graph G its distance matrix D is defined as follows.

$$D[i,j] = \begin{cases} 1, & \text{if (i,j) is an edge of G} \\ d, & \text{d is the length of the shortest path [i–j]} \\ \infty, & \text{if no such path exists} \end{cases}$$

Note that D is symmetric and the adjacency matrix A is contained in D. The distance matrix has several applications. For instance, the radius, diameter and the center of a graph are defined in terms of its distance matrix. The distance matrix has also been used in for detecting graph isomorphism.

The problem of updating the distance matrix involves recomputing D for the modified graph that is obtained from the original graph after a minor change has been made to it. Cheston [1] has studied sequential algorithms for updating the distance matrix. Our algorithms for updating the distance matrix on a PRAM require O(log n) time and use $O(n^3)$ processors. The start-over algorithm for initial computation of D requires $O(\log^2 n)$ time and use $O(n^3)$ processors. Our algorithms therefore are efficient when compared to the start-over algorithm.

To design efficient parallel algorithms for updating the distance matrix we proceed as follows. The first step is to determine the vertex pairs whose distances are unaffected by the graph change. In particular, we need to compute these pairs after an edge has been deleted from G. Note that the i^{th} row of D corresponds to a DT (distance tree) for G, that is rooted at vertex i. Our update algorithms require that n such DTs for G, one for each row of D, be stored along with the matrix D. We speed up the computation by storing these trees as inverted trees. Now, an edge deletion operation may create two subtrees out of a single DT. Using the techniques described earlier (see Section 2), we can identify the vertices that belong to each of these subtrees in O(log n) time. Therefore we can determine the entries of D that are unaffected by the graph change. After the matrix D has been recomputed we can reconstruct these DTs as inverted trees.

The other cases of edge and vertex insertion can be handled without DTs. We do not consider the problem of vertex deletion, since deletion of a vertex from a tree may split it into more than two subtrees. We are unable to handle this situation using our approach.

In order to describe the actual computational steps of our algorithms and the proof of their correctness, we first describe the parallel start-over algorithm for computation of the distance matrix.

Start-Over Algorithm

It has been observed in [2] that distances for all pairs of vertices in a graph (directed or undirected) can be computed in $O(\log^2 n)$ time on a PRAM by straightforward parallelization of the known sequential algorithm that is based on repeated multiplication of the adjacency matrix. In this parallel algorithm addition and minimization operations replace multiplication and addition operations of an inner product step. We refer to this as the *plus-min* multiplication of two matrices. The algorithm initializes the distance matrix D to the adjacency matrix A and then performs (log n) iterations of the plus-min multiplication of D by itself. The matrix DD is used as temporary storage in the algorithm for clarity.

```
// All steps involving i and j are executed for all i, j $1 \le i \le n$ and $1 \le j \le n$ //
1.   D[i,j] := A[i,j]     //Initialize//
2.   for t:=1 to log(n-1) do
2a.      DD[i,j] := min { D[i,j], D[i,k] + D[k,j] } // $1 \le k \le n$ $i \ne k$ $j \ne k$//
2b.      D[i,j] := DD[i,j]
```

Algorithm 3.1

Lemma 3.1: The above algorithm computes the distance matrix D in $O(\log^2 n)$ time using $O(n^3)$ processors.

Proof: Steps (1) and (2b) can be done in constant time using n^2 processors. Step (2a) can be done in O(log n) time by assigning n processors to compute each element of the matrix DD. Since DD has n^2 elements we need $O(n^3)$ processors to perform step (2a). Note that at the end of t^{th} iteration we would have found distances for those pairs whose vertices are at most 2^t units apart. Since the maximum distance for any pair of vertices is at most (n-1) units, we need log(n-1) iterations of step (2a).

Construction of Distance Trees from D

We now describe the computational steps for constructing n DTs, each rooted at a different vertex. Let T_i denote a DT that is rooted at vertex i. Recall that the function F completely specifies an inverted tree (see Section 2). Let the function F_i completely specify T_i. F_i is computed as follows.

1. Set $F_i(i) = i$, because i is the root of the tree.
2. For every vertex in T_i other than its root, determine its father in T_i. This can be done in O(log n) time using n processors for a vertex. Consider vertex j and let its distance from the root be d_j. Therefore the father of j must be a vertex that is (d_j-1) units away from the root. Assign n processors to j and select a vertex k such that k is adjacent to j and (d_j-1) units away from i. Select a minimum k to break the ties. Set $F_i(j) = k$. Since j can have at most n neighbors this minimization needs O(log n) time and n processors. As there are n vertices in each tree we need $O(n^2)$ processors to construct T_i. Therefore we have the following lemma.

Lemma 3.2 The construction of n DTs from the matrix D requires O(log n) time and uses $O(n^3)$ processors.

We now proceed to describe our update algorithms and the proof of their correctness.

Edge deletion

The problem of edge deletion update is concerned with recomputing the distance matrix D after an edge has been deleted from the graph. In order to recompute D, we first identify the pairs of vertices whose distances are unaffected by the edge deletion step. We then construct matrix D′ such that

$$D'[i,j] = \begin{cases} D[i,j], & \text{if } D[i,j] \text{ is unchanged.} \\ \infty & \text{otherwise} \end{cases}$$

Now, two iterations of steps (2a) and (2b) of Algorithm 3.1 on D′ recompute the distance matrix for the new graph. We will show later on that two iterations are sufficient for recomputing D. The computational steps are described for tree T_i. There are n such trees and the following steps for T_i are executed in parallel for all of them.

Let (x,y) be the edge that was deleted from G. Also assume that the T_i's have been constructed from D.

1. If (x,y) is not in T_i then the i^{th} row of D is not affected at all. This is so because T_i stays connected even after (x,y) has been deleted from G. Therefore the distances to all other vertices from i in the new graph do not change.

   Now suppose that the edge (x,y) is in T_i. Assume, without loss of generality that its direction in T_i is from x to y. To delete (x,y) set $F_i(x) = x$. This creates a forest of two subtrees one of which is rooted at i and the other at x.
2. Identify the vertices that belong to different subtrees. The vertices in the subtree rooted at x are not reachable from i in T_i. Therefore their distances from i must be recomputed. By Lemma 2.1, the two subtrees can be identified in O(log n) time and requires $O(n^2)$ processors.
3. Compute the i^{th} row of D′ as follows. The distances to the vertices that are not reachable from i are marked as ∞. For other vertices that are reachable from i, the distances have not changed. This step can be done in constant time using $O(n^2)$ processors.
4. Perform two iterations of the start-over algorithm on D′ to compute the updated distance matrix. This computation requires O(log n) time and uses $O(n^3)$ processors

We now prove the correctness of our algorithm.

Theorem 3.1: Two iterations of the start-over algorithm that operates on D' are sufficient to compute the updated distance matrix for the new graph.

Proof: Consider the tree rooted at i (see Fig. 3.1). Let (x,y) be the edge that was deleted. If (x,y) is a bridge of G then (x,y) is present in all DTs, and hence in T_i. Deletion of (x,y) disconnects G as well as all T_i's. The vertices in the subtree rooted at x are not reachable from i in G. Therefore D' itself is the new distance matrix.

On the other hand if (x,y) is not a bridge of G then there exists at least one edge in G that connects these two subtrees that were created by deletion of (x,y). Let u be a vertex in the subtree rooted at x that has such an edge incident on it. Since T_i is a DT for G, u can have neighbors that are at a distance d_u, d_u+1 or d_u-1 (where d_u is the distance to u from i). Therefore at the end of the first iteration (i.e., *plus-min* multiplication of D' by itself) we would have found the distances to all such u's from i.

Now consider a vertex v that does not have a neighbor in the subtree rooted at i. The shortest path from i to v must pass through some such u (a vertex that has a neighbor in the subtree rooted at i). Therefore the distance to v from i can be expressed as the sum of distances, one from i to u and the other from u to v. Now, at the end of the second iteration we would have computed the distances to all such v's from i. This requires that $D'[u,v]$ must not have been marked as ∞ in step (3) of our algorithm. In other words, v must be reachable from u in the DT T_u even after the edge (x,y) has been deleted from G. We now show that this indeed is the case !

The subtree rooted at x contains a path [u-v] that passes through their lowest common ancestor. The length of this path is $(d_u+d_v-2d_w)$, where w = LCA(u,v). This path is shorter than any other path [u,v] that contains the edge (x,y), because the length of such

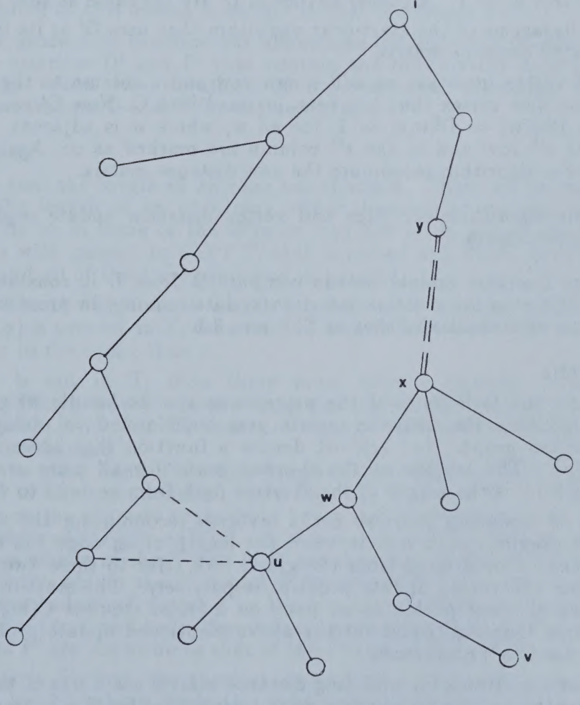


Fig. 3.1

a path is at least $(d_u+d_v-2d_y)$ and $d_y<d_w$ (note that y is an ancestor of w). Therefore the DT T_u contains the shortest path from u to v that is either a concatenation of paths [u-w] and [w-v], or a path shorter than this, which does not contain (x,y). Hence the distance D[u,v] is not affected by deletion of (x,y).

It is possible that vertex v is reachable from many such u's, but the minimization operation will select a vertex that minimizes the length of the path from i to v. Therefore at the end of the second iteration we would have computed the shortest paths to all vertices from the root i. Hence the theorem.

The ideas described in this proof will be used in subsequent sections to outline the similar proofs.

Theorem 3.2: Our algorithm updates the distance matrix of an undirected graph after an edge deletion operation in O(log n) time and use $O(n^3)$ processors.

Proof: By Lemma 3.2, the construction of the DTs needs O(log n) time and $O(n^3)$ processors. The computations on an inverted tree require O(log n) time and $O(n^2)$ processors. As there are n such trees, we need $O(n^3)$ processors for step (2). The matrix D′ can be constructed from D in constant time using $O(n^2)$ processors. Now two iterations of the start-over algorithm are sufficient to compute the new distance matrix. By Lemma 3.1, each iteration needs O(log n) time and uses $O(n^3)$ processors.

Edge and Vertex Insertion

We now describe our algorithms for updating the distance matrix after an edge or a vertex has been inserted into G. In order to compute D′ from D after an edge insertion operation we proceed as follows. Let (u,v) be the edge that has been inserted into G.

1. Set $D'[u,v] = D'[v,u] = 1$. All other entries of D′ are the same as that of D.
2. Perform two iterations of the start-over algorithm that uses D′ as its input to compute the updated distance matrix.

In the case of vertex insertion we add a new row and a column to the old distance matrix. Let z be the new vertex that has been inserted into G. Now D′ can be obtained from D by setting $D'[z,w] = D'[w,z] = 1$, for all w, where w is adjacent to z. All the other entries in the z^{th} row and in the z^{th} column are marked as ∞. Again, two iterations of the start-over algorithm recompute the new distance matrix.

Theorem 3.3: Our algorithms for edge and vertex insertion update require O(log n) time and use $O(n^3)$ processors.

Proof: For an edge insertion update we can compute D′ from D in constant time using one processor and this step for a vertex insertion update requires 2n processors. The rest of this proof is along lines similar to that of Theorem 3.1.

4. Shortest Paths

We now extend the techniques of the previous section to handle weighted graphs. The problem of updating the distance matrix gets transformed to updating shortest paths in an undirected graph. Let $L:E\rightarrow R$ denote a function that associates a length with the edges of G. The lengths of the shortest paths for all pairs are stored in a matrix P such that P[i,j] is the length of the shortest path from vertex i to vertex j.

The problem of updating shortest paths involves recomputing the shortest path matrix P from the previous such matrix when the length of an edge has changed or a vertex has been inserted or deleted from the graph. We refer to these two problems as the edge update and the vertex update problem respectively. The start-over algorithm in [2] for computing shortest paths for all pairs on a PRAM requires $O(\log^2 n)$ time and uses $O(n^3)$ processors. Our algorithms for the above mentioned update problems require O(log n) time and use $O(n^3)$ processors.

Recall that our algorithms for updating distance matrix use n DTs of the graph. For updating shortest paths we employ shortest path trees (SPTs) for this purpose. Since we are dealing with weighted graphs, the shortest path matrix P cannot be used to construct the SPTs. In order to construct SPTs using the technique for constructing the DTs, we actually need the number of edges on all shortest paths. Therefore, the start-over algorithm described in the previous section needs to be modified to compute this. This information is stored in the matrix D wherein D[i,j] is the number of edges on the shortest path from i to j. The SPTs that are constructed from D are maintained as inverted shortest path trees.

We now describe the start-over algorithm for computing the shortest paths that has been modified to compute D.

Start-Over Algorithm

As described earlier, the parallel algorithm for computing shortest paths for all pairs is based on the repeated plus-min multiplication of matrix P that has been initialized to edge-lengths. The computational steps are as follows.

```
//All steps involving i and j are executed for all i,j   $1 \le i \le n$   $1 \le j \le n$//
1.    P[i,j] := L(i,j)    //the length of edge (i,j)//
      D[i,j] := A[i,j]    //A is the adjacency matrix.//
2.    for t := 1 to log(n-1) do
2a.       PP[i,j] := min { P[i,j], P[i,k] + P[k,j] }  //$k \neq i$ and $k \neq j$//
2b.       DD[i,j] := D[i,k] + D[k,j]  //for the same k obtained in the step 2a//
2c.       P[i,j] := PP[i,j];  D[i,j] := DD[i,j]
```

Algorithm 4.1

Lemma 4.1: The above algorithm computes the shortest path matrix P in $O(\log^2 n)$ time and requires $O(n^3)$ processors.

Proof: The proof of this lemma is similar to that of Lemma 3.1.

We now proceed to describe our algorithms for updating shortest paths. First, we compute the matrices D' and P' that contain the information unaffected by the incremental change. Now two iterations of the start-over algorithm on D' and P' are sufficient to recompute D and P for the new graph.

Edge Update

Assume that the length of an edge has changed. There are several cases to be handled. First, the length of an edge may either decrease or increase and this edge may either currently be in some of the SPTs or may not be in any of the SPTs. We describe our algorithm with respect to a SPT T_i that is rooted at i. Note, however, that there are n such trees and all of them are processed in parallel.

Let (x,y) be the edge whose length has decreased. Assume, without loss of generality that if (x,y) is present in T_i then its direction is from x to y. If (x,y) is not in T_i then let y be closer to the root i than x.

1. If (x,y) is not in T_i then there must exist a shortest path [x-y] such that $L(x,y) \geq P[x,y]$. Now if the new length of (x,y) is still greater than or equal to that of the path [x-y] then T_i does not change. Therefore the i^th^ row of D and P remains unchanged.

   On the other hand if the new $L(x,y) < P[x,y]$ then we have a shorter path from i to x and to all descendants of x (see Fig. 4.1). Let $\delta = D[x,y] - 1$. Now the number of edges on the shortest paths from i to x and to descendants of x have reduced by δ. Therefore set $D'[i,j] = D[i,j] - \delta$, where j is either x or a descendant of x. The remaining entries of the i^{th} row of D' are the same as that of i^{th} row of D.

   Similarly, obtain the i^{th} row of P' by setting $P'[i,j] = P[i,j] - \theta$, where j is either x or a descendant of x and $\theta = P[x,y] - L(x,y)$. Here, the lengths of the shortest paths from i to x and to its descendants have reduced by θ. The other entries of the i^{th} row of P' are the same as that of the i^{th} row of P.

2. If (x,y) is in T_i then the shortest paths from i to x and to its descendants have reduced by Δl, where Δl is the reduction in the length of (x,y). In this case D remains unchanged but the i^{th} row of P is affected. Therefore set $P'[i,j] = P[i,j] - \Delta l$, where j is either x or a descendant of x.

3. Perform one iteration of the start-over algorithm to recompute the matrices D and P for the new graph.

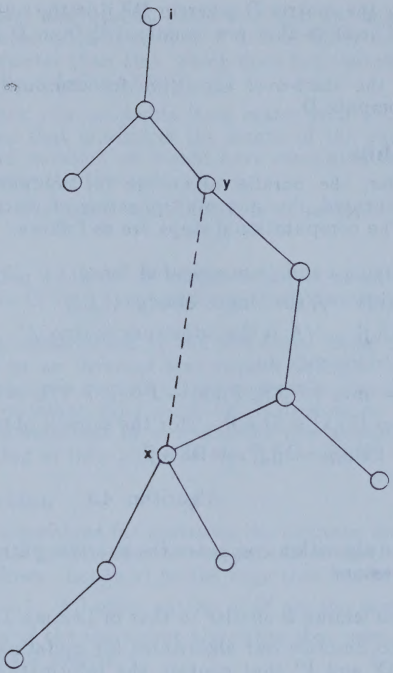


Fig. 4.1

Now suppose that the length of (x,y) has increased. If (x,y) is not in T_i then none of the distances from i to other vertices change. Hence the i^{th} row of D and P are unaffected. If (x,y) is in T_i then set $P'[x,y]$ to the new length of (x,y) and the rest of P' is the same as P. The matrix D is not affected at all. In this case two iterations of the start-over algorithm on D' and P' update the matrices D and P.

Theorem 4.1: Our algorithm for the edge update problem requires O(log n) time and uses $O(n^3)$ processors.

Proof: By Lemma 2.1, the descendants of x can be determined in O(log n) time using $O(n^2)$ processors. All other steps except the iterations of the start-over algorithm can be done in constant time using $O(n^2)$ processors. As there are n such SPTs we need $O(n^3)$ processors. By Lemma 4.1, one iteration of the start-over algorithm can be done in O(log n) time using $O(n^3)$ processors. The proof of the sufficiency of two iterations for recomputing D and P is along the same line as that of Theorem 3.1.

Note that our algorithm can easily handle the cases where an edge is inserted or deleted from the graph, since we can treat the nonexistent edges as edges having infinite weights.

Vertex Update

The problem of vertex update deals with recomputing the shortest paths after a vertex has been inserted into the graph. We add a new row and a column to D' and P' for the new vertex z. Now set $D'[w,z] = D'[z,w] = 1$ and $P'[w,z] = P'[z,w] = L(z,w)$, where w is a neighbor of z. All other entries of z^{th} row and z^{th} column of D' and P' are respectively set to 0 and ∞. Now execute the start-over algorithm on D' and P'. Two

Performance of Entropy-Constrained Block Transform Quantizers[1]

by
N. Farvardin and F. Y. Lin
Electrical Engineering Department
and
Systems Research Center
University of Maryland
College Park, Maryland 20742

Summary

There are numerous techniques available for analog-to-digital conversion,' or data compression, of discrete-time processes. They range from simple scalar (zero-memory) quantization or PCM, to more sophisticated schemes such as predictive encoding, tree encoding and multi-dimensional quantization. It is well-known that the potential gains of data compression is most noticeable for highly correlated sources.

One of the most popular data compression schemes for encoding of correlated sources is block transform quantization. The main advantages associated with the block transform quantization scheme are: (i) good performance results, and (ii) ease of implementation.

In this paper we study the rate-distortion theoretic performance of optimal block transform quantization schemes on first-order stationary autoregressive processes. More precisely, we assume that blocks of length L of source outputs are operated upon by the Karhunen-Loeve transformation. The transform coefficient are then quantized and coded by means of *optimal entropy-constrained* zero-memory quantizers operating at rates $H_1, H_2, \ldots, H_L$. The problem that we address is that of determining these rates in such a way that the overall average squared-error distortion is minimized.

This problem, which is very much similar to that studied by Huang and Schultheiss [1], can be formulated as the following constrained nonlinear programming problem:
Minimize

$$D(\underline{H}) = \frac{1}{L}\sum_{i=1}^{L} \lambda_i D_i(H_i) \, , \tag{1.a}$$

subject to

$$\frac{1}{L}\sum_{i=1}^{L} H_i \le R \, , \tag{1.b}$$

in which $\underline{H} = [H_1, H_2, \ldots, H_L]$, λ_i is the variance of the ith coefficient and $D_i(H)$ is the variance-normalized distortion-rate performance of the optimum entropy-constrained

---

[1] This work was supported in part by a grant from Martin Marietta Laboratories, in part by a grant from the Minta Martin Fund for Aeronautical Research and in part by the General Research Board of the University of Maryland.

[4] S. Fujishige, "A Note on the Problem of Updating shortest Paths", *Networks 11* (1981), pp. 317-319.

[5] D. Hirschberg, "Parallel Algorithms for the Transitive Closure and the Connected Component Problem", *Proc. Eighth STOC,* 1976, pp. 55-57.

[6] D. Hirschberg, A. K. Chandra and D. V. Sarwate, "Computing Connected Components on Parallel Computers", *CACM 22* (1979), pp. 461-464.

[7] S. Pawagi and I. V. Ramakrishnan, "An O(log n) Algorithm for Parallel Update of Minimum Spanning Trees", TR 1452, Dept. of Computer Science, Univ. of Maryland, 1984. Also to appear in Information Processing Letters.

[8] S. Pawagi and I. V. Ramakrishnan, "Parallel Update of Graph Properties in Logarithmic Time", *Fourteenth International Conference on Parallel Processing,* 1985, pp. 186-193.

[9] S. Pawagi and I. V. Ramakrishnan, "On Using Multiple Inverted Trees for Parallel Updating of Graph Properties", TR 1502, Dept. of Computer Science, Univ. of Maryland, 1985.

[10] C. Savage, "Parallel Algorithms for Some Graph Problems", TR 784, Dept. of Mathematics, Univ. of Illinois, Urbana, 1977.

[11] C. Savage and J. Ja'Ja', "Fast Efficient Parallel Algorithms for Some Graph Problems", *SIAM J. Comp. 10* (1981), pp. 682-691.

[12] P. Spira and A. Pan, "On Finding and Updating Spanning Trees and Shortest Paths", *SIAM J. Comp. 4* (1975), pp. 375-380.

[13] Y. Tsin and F. Chin, "Efficient Parallel Algorithms for a Class of Graph-Theoretic Problems", *SIAM J. Comp. 14* (1984), pp. 580-599.

A FAMILY OF MERGING NETWORKS DERIVED FROM THE BITONIC MERGER

GIANFRANCO BILARDI
Department of Computer Science
Cornell University, Ithaca, NY 14853

ABSTRACT

A class of transformations is defined that can be applied to the bitonic network without affecting its merging capabilities. One of the networks obtained from these transformations is the balanced merger.

1. INTRODUCTION

The bitonic merger (*BM*) is a classical network of comparator-exchangers (*CEs*) capable of merging two sorted sequences [Ba 68]. Here, we define a class of transformations that, when applied to the *BM*, yield new merging networks. These transformations are simple, but not trivial, in the sense that they can not be viewed as a mere renaming of the lines of the bitonic network.

An interesting finding is that the balanced network, introduced in [DPRS 83] as a novel merging network, is indeed a member of the family of networks generated by our transformations, and is therefore more closely related to Batcher's network than originally thought.

If, in the *BM*, *CEs* are replaced by switches (*SWs*) the result is the Ω permutation network [La 75]. Moreover, the permutations that are sorted by the *BM* are exactly those that can be realized by the Ω network (Ω permutations). This fact makes possible and indeed fruitful to study the bitonic and the Ω networks in the same framework. In this work, we exploit some algebraic results of [St 83] on the structure of the class of permutations Ω.

Besides the already mentioned references, [Kn 73], [Pe 77], [HS 82] and [Pr 83] contain material closely related to our subject.

2. DEFINITIONS AND NOTATION

Following [Kn 73], we view a *network* as a set of lines (which can be imagined to run parallel, horizontally) connected by a set of devices (which can be imagined as vertical connections between pairs of lines). The devices are organized in a series of stages, with no two devices in a stage sharing a line. The devices we shall consider have two inputs, a and b, and two outputs, c and d, and are of one of two types: *comparator-exchanger* (CE) and *switch* (SW). In a CE, $c = \min(a,b)$ and $d = \max(a,b)$. In a SW, there is also a binary state s that can be set independently of the inputs; when $s=0$, $(c,d) = (a,b)$, and when $s=1$, $(c,d) = (b,a)$. A CE can be viewed as a SW whose state is a function of the inputs ($s=0$ *iff* $a<b$). In the sequel, when we say that a device connects the pair of lines (i,j), we mean that a and c are on line i, b and d on line j.

The output of a network of either CEs or SWs is a *permutation* of the input. More precisely, if the network has N lines numbered from 0 to $N-1$, and we let X_i and Y_i be respectively the input and the output value at line i, then $Y_{\pi(i)} = X_i$, where $\pi = (\pi(0),...,\pi(N-1))$ is a permutation of $(0,...,N-1)$.

It is convenient to define the *series* and the *parallel* connections of two networks NT_1 and NT_2 of N_1 and N_2 lines, respectively. The series of NT_1 and NT_2 is obtained by connecting the *i-th* output line of NT_1 to the *i-th* input line of NT_2, and is defined only when $N_1 = N_2$. We denote it by NT_2NT_1. The parallel of NT_1 and NT_2 has N_1+N_2 lines, the first N_1 of which connected as in NT_1 and the last N_2 of which connected as in NT_2. We denote it by $NT_1 \bigcirc NT_2$.

If NT_r realizes permutation π_r ($r=1,2$), then NT_2NT_1 realizes the product $\pi_2\pi_1$, and $NT_1 \bigcirc NT_2$ realizes $\pi_1 \bigcirc \pi_2$ defined as $(\pi_1 \bigcirc \pi_2)(i) = \pi_1(i)$ for $i=0,...,N_1-1$, and $(\pi_1 \bigcirc \pi_2)(i+N_1) = \pi_2(i)+N_1$ for $i=0,...,N_2-1$. Note that $\bigcirc$ is associative.

We are now ready to define the bitonic merger BM_n and the omega network OM_n on $N=2^n$ lines.

Definition 1. Let C_n ($n \geq 1$) be a one-stage network on $N=2^n$ lines with a CE between the pair $(i,i+N/2)$, for $i=0,...,N/2-1$. Then, BM_n is recursively defined as $BM_n = (BM_{n-1} \bigcirc BM_{n-1})C_n$, for $n>1$, and $BM_1 = C_1$.

Definition 2. Let $W_n (n \geq 1)$ be a one stage network on $N=2^n$ lines with a SW between the pair $(i,i+N/2)$, for $i=0,...,N/2-1$. Then, OM_n is recursively defined as $OM_n = (OM_{n-1} \bigcirc OM_{n-1})W_n$, for $n>1$, and $OM_1 = W_1$.

Definition 3. Ω_n is the class of permutations realizable by OM_n.

The class Ω_n has been introduced by [La 75] who has given a characterization of the Ω permutations, later reformulated more formally in [Pe 77]. Another characterization is given in [Pr 83], where the results are actually expressed on the class of permutations sorted by BM_n. How we shall see, the latter class equals Ω_n.

We denote by ι_n the identity permutation, and by σ_n the swap permutation $\sigma_n = (\sigma_n(0),...,\sigma_n(N-1)) = (N/2,...,N-1,0,...,N/2-1)$, which is realized by W_n when all the SWs are in state 1.

The following two classes of permutations are central to our network transformations.

Definition 4. The class L_n of the *lower triangular* permutations is recursively defined as

$$L_n = \{(\alpha \bigcirc \alpha)\delta \mid \alpha \in L_{n-1}, \delta \text{ is realizable by } W_n\},$$

for $n>1$, with L_1 containing both permutations of $(0,1)$.

Definition 5. The class M_n of the *upper triangular* permutations is recursively defined as

$$M_n = \{(\alpha \bigcirc \beta)\delta \mid \alpha, \beta \in M_{n-1}, \delta \in \{\iota_n, \sigma_n\}\},$$

for $N>1$, with M_1 containing both permutations of $(0,1)$.

Lower and upper triangular permutations have been introduced in [Pe 77], with definitions different but equivalent to ours. Steinberg [St 83] has shown that L_n and M_n contain exactly the right invariant and the left invariant permutations of Ω_n. More precisely:

Proposition 1. $\lambda \in L_n$ if and only if, for all $\pi \in \Omega_n$, $\pi\lambda \in \Omega_n$.

Proposition 2. $\mu \in M_n$ if and only if, for all $\pi \in \Omega_n$, $\mu\pi \in \Omega_n$.

3. TRANSFORMATIONS OF THE BITONIC MERGER

Without loss of generality, we shall assume that the input N-tuple of a network of either CEs or SWs is a permutation $\pi = (\pi(0),...,\pi(N-1))$ of $(0,...,N-1)$. We say that a network of CEs sorts π if the output corresponding to π is the identity (line i outputs i, for $i=0,...N-1$). We say that a network of SWs passes π if there exists a setting of the state of the SWs such that the output corresponding to π is the identity. Thus, $\pi(i)$ can be viewed as the destination of the *i-th* input.

An interesting property of OM_n is that there is a unique path from a given input i to a given output j. Let i and j respectively have the binary expansions $i_{n-1}i_{n-2}...i_1i_0$ and $j_{n-1}j_{n-2}...j_1j_0$. It is easily shown that, after k stages, the path from i to j goes through line $j_{n-1}...j_{n-k}i_{n-k-1}...i_0$ [La 75]. Thus, in any SW, the input with larger destination will go to the higher numbered output line, exactly as in a CE. This property, together with the observation that, when SWs are replaced with CEs, OM_n becomes BM_n, leads to the following interesting fact.

Proposition 3. A permutation π is in Ω_n if and only if is sorted by BM_n.

Definition 6. A permutation π is *bitonic* if there exists a cyclic shift of π consisting of an increasing sequence followed by a decreasing one.

Batcher has shown that all the bitonic permutations are sorted by the BM and therefore, by Proposition 3, are in Ω_n. When the BM is used as a merger, its input is obtained by concatenating, in opposite order, the two (sorted) sequences to be merged.

We now extend this result by showing that there are many other input protocols (besides that assigning $\pi(i)$ to input line i) which guarantee that a bitonic permutation is sorted by a bitonic network.

Definition 7. Given a permutation λ and an input sequence π, we denote by $BM_n(\lambda)$ the bitonic merger with the input protocol prescribing that the *i-th* line receives the value $\pi(\lambda(i))$, for $i=0,\ldots,N-1$

Theorem 1. If $\lambda \in L_n$, then $BM_n(\lambda)$ sorts all permutations in Ω_n.

Proof. If π is in Ω_n, and $\lambda \in L_n$ then, by Proposition 1, then $\pi\lambda \in \Omega_n$, and , by Proposition 3, $\pi\lambda$ is sorted by BM_n.

It is easy to show that $| L_n| = 2^{N-1}$ $(N=2^n)$. Thus, Theorem 1 provides us with a class, 2^{N-1} merging networks, $\{BM_n(\lambda) \mid \lambda \in L_n\}$. The transformation of input protocol is a very simple one, and does not even affect the structure of the network. However, when combined with a transformation of the kind to be described next, it may lead to networks that (superficially) bear little resemblance with BM_n.

Another way of modifying the BM consists in changing the direction of a suitable subset of *CEs*. This subset is uniquely identified by the permutation ω realized by the network when the input is the identity. This fact is an immediate consequence of the uniqueness of the path between any fixed pair of input and output lines. The next proposition states that there is a one-to-one correspondence between subsets of *CEs* and responses to the identity.

Proposition 4. For every $\omega \in \Omega_n$, there exists a set $S(\omega)$ of *CEs* in BM_n, such that, by changing their direction, one obtains a network $\omega \sim BM_n$ that, on the identity input, realizes ω.

Proof. $S(\omega)$ corresponds to the set of *SWs* in OM_n that output the larger of their input values on the lower numbered line. when OM is in the configuration that realizes ω, and line i inputs value i, for $i=0,\ldots,N-1$.

Of particular interest is the case of networks obtained from upper triangular permutations, which enjoy the following property.

Proposition 5. If $\mu \in M_n$, then $\mu \sim BM_n$ yields the same output for every input $\pi \in \Omega_n$. In other words, $\mu \sim BM_n$ realizes the permutation $\mu\pi$ for each $\pi \in \Omega_n$.

Thus, since $\iota_n \in \Omega_n$, for every input $\pi \in \Omega_n$, $\mu \sim BM_n$ outputs $\mu^{-1}(i)$ on line i $(i=0,\ldots,N-1)$. We can then rename the lines of $\mu \sim BM_n$ so that its output will appear sorted.

Theorem 2. Let $\mu \in M_n$, and let $\mu\text{-}BM_n(\mu)$ the network obtained from $\mu \sim BM_n$ by renaming line i as $\mu^{-1}(i)$, with the input protocol prescribing that line j (of $\mu\text{-}BM_n(\mu)$) inputs the $\mu(j)$*-th* element of the input permutation. Then, $\mu\text{-}BM_n(\mu)$ sorts all $\pi \in \Omega_n$.

With arguments similar to those used in Theorem 1, the previous theorem can be generalized as follows.

Theorem 3. If $\mu \in M_n$ and $\lambda \in L_n$, then $\mu\text{-}BM_n(\lambda\mu)$ sorts every $\pi \in \Omega_n$.

The networks $\{\mu\text{-}BM_n(\lambda\mu) \mid \mu \in M_n, \lambda \in L_n\}$ form the family of mergers referred to in the title of this paper.

4. THE BALANCED MERGER

The *balanced merger*, introduced by [DPRS 83], can be defined as follows.

Definition 8. Let $D_n (n \geq 1)$ be a one-stage network on $N=2^n$ lines with a CE between the pair $(i, N-1-i)$, for $i=0,\dots,N/2-1$. Then, BL_n is recursively defined as $BL_n = (BL_{n-1} \bigcirc BL_{n-1}) D_n$, for $n>1$, and $BL_1 = D_1$.

In [DPRS 83] it is shown that BL_n merges two sorted sequences of $N/2$ elements each, when one sequence is input on the even-numbered lines, and the other on the odd-numbered ones. In the sequel, we show that BL_n is in the family of networks defined in the previous section.

Definition 9. Let $N=2^n$, $(n \geq 1)$. We define the permutation τ_n of $(0,\dots,N-1)$ by the relation $j_k = i_k \oplus i_{k+1}$ $(k=0,\dots,n-1)$, where $i_{n-1}\dots i_0$ and $j_{n-1}\dots j_0$ are the binary representations of i and $\tau_n(i)$, respectively, and $i_n = 0$.

It is not difficult to show that $\tau_n \in M_n$. The next proposition due to [St 83] establishes an interesting one-to-one correspondence between upper triangular and lower triangular permutations.

Proposition 6. Let ρ be the *bit-reversal* permutation such that the binary representation of $\rho(i)$ is $i_0 \dots i_{n-1}$, where $i_{n-1}\dots i_0$ is the representation of i. Then. $\rho_n \mu \rho_n \in L_n$ if and only if $\mu \in M_n$.

It is easily seen that L_n and M_n are groups. Thus, if $\mu \in M_n$, then $\mu^{-1} \in M_n$, and $\rho_n \mu^{-1} \rho_n \in L_n$, by Proposition 6. In Particular, we are interested in $\rho_n \tau_n^{-1} \rho_n$, which satisfies the following property.

Proposition 7. The binary representations $j_{n-1}\dots j_0$ and $i_{n-1}\dots i_0$ of j and $i=\rho_n \tau_n^{-1} \rho_n(j)$ are related by $i_k = j_k \oplus \dots \oplus j_0$.

The next theorem states that, if $\mu=\tau_n$ and $\lambda=\rho_n \tau_n^{-1} \rho_n$, then $\mu\text{-}BM_n(\lambda\mu)$ is the balanced network.

Theorem 4. Let $\xi_n = \rho_n \tau_n^{-1} \rho_n \tau_n$. Then, $BL_n = \tau_n\text{-}BM_n(\xi_n)$.

Sketch of proof. (Input protocol) Let $i_{n-1}\dots i_0, j_{n-1}\dots j_0$, and $l_{n-1}\dots l_0$ respectively be the binary representations of i, $j=\tau_n(i)$ and $\xi_n(i)=\rho_n \tau_n^{-1} \rho_n(j)$. From Propositions 7 and 6, $l_k = j_k \oplus j_{k-1} \oplus \dots \oplus j_0 = i_{k+1} \oplus i_k \oplus i_k \oplus i_{k-1} \oplus \dots \oplus i_1 \oplus i_0$, which can be simplified to $l_k = i_{k+1} \oplus i_0$. If i is even, $i_0 = 0$, and $l_{n-1} \ldots l_0 = 0 i_{n-1} \ldots i_1$, so that $\xi_n(i) = i/2$. If i is odd, $i_0 = 1$, and $l_{n-1} \ldots l_0 = 1 \overline{i}_{n-1} \cdots \overline{i}_1$, so that $\xi_n(i) = N+1-2i$. Thus, $\xi_n(i)$ is the index of the element of a bitonic sequence to be input at line i in the balanced network.

(Comparators) By definition, τ_n-BM_n is obtained from BM_n by changing the direction of some comparators, and then renaming line i as $j = \tau_n^{-1}(i)$. Inversion of the equation given in Definition 9 yields $j_{n-k} = i_{n-1} \oplus \ldots \oplus i_{n-k}$. If i and h have the same binary representation except for the k-th bit, and $i<h$, then the pair of lines (i,h) is connected by a *CE* in BM_n. Thus, $\tau_n^{-1}(i)$ and $\tau_n^{-1}(h)$ are connected by a *CE* in τ_n-BM_n. But $|\tau_n^{-1}(h) - \tau_n^{-1}(i)| = 2(i \bmod 2^k)$, how it must be for lines connected by a *CE* in the balanced network. A more elaborate argument will establish that also the direction of the *CEs* is the same in both networks.

5. CONCLUSION

Theorem 1 shows 2^{N-1} ways to input two sorted sequences in the bitonic network and obtain a sorted output. Theorem 2 shows 2^{N-1} ways to change the direction of the comparators and permute the lines in the bitonic network that lead to networks that can still sort all omega permutations. Theorem 3 combines these results to define a class of merging networks, all of which work essentially on the same principle: form an omega permutation from two sorted sequences. Theorem 4 relates the balanced network to the bitonic one.

At this stage, it is not clear to us whether any of the networks in our class has any special feature that would make it preferable, say, to the bitonic network. For example, [DPRS 83] have shown that the series of $\log N$ balanced mergers is a sorter. This property does not hold for the bitonic merger (although the property holds if the outputs of one stage are permuted by τ_n before being fed into the next stage). However, we think that this study contributes to a better understanding of the bitonic and the omega networks, and indicates that it may be worthwhile to investigate comparator networks in connection with the corresponding switching networks.

REFERENCES

[Ba 68] K.E. Batcher, "Sorting networks and their applications," in *Proceedings AFIPS Spring Joint Comp. Conf.*, Vol. 32, pp. 307-314, 1968.

[DPRS 83] M. Dowd, Y. Perl, L. Rudolph, and M. Saks, "The balanced sorting network," in *Proceedings ACM Princ. Distrib. Comput.*, pp.161-172, August 1983.

[HS 82] Z. Hong, R. Sedgewick, "Notes on merging networks," *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pp.296-302, San Francisco, CA, May 1982.

[Kn 73] D.E. Knuth, "The art of computer programming," *Vol. 3: Sorting and Searching*, Addison Wesley, Reading, MA, 1973.

[La 75] D. Lawrie, "Access and alignment in an array processor," *IEEE Trans. Comput.*. vol. C-24, No.12, p. 1145-1155, December 1975.

[Pe 77] M.C. Pease, "The indirect binary n-cube microprocessor array," *IEEE Trans. Comput.*, vol. C-26, no.5, pp. 458-473, May 1977.

[St 83] D. Steinberg, "Invariant properties of the shuffle-exchange and a simplified cost-effective version of the omega network," *IEEE Trans. Comput.*, vol. C-32, no.5, pp.444-450, May 1983.

SIMULATIONS BETWEEN MESH-CONNECTED PROCESSOR ARRAYS[†]

Mikhail J. Atallah
Dept. of Computer Science, Purdue Univ., West Lafayette, IN. 47907

When a mesh U can simulate every step of another mesh W with $O(h)$ of its own steps, we say that U *h-simulates* W. Note that if U h-simulates W then any problem that W solves in time T can be solved by U in time $O(h.T)$. Let S denote a square d-dimensional mesh of n processors, i.e. S is an $n^{1/d}\times\cdots\times n^{1/d}$ mesh. Let R denote a rectangular d-dimensional mesh of n processors, i.e. R is an $l_1\times l_2\times\cdots\times l_d$ mesh where $l_1.l_2.\cdots.l_d=n$. Then we have the following.

Theorem 1. Mesh R can $\max(l_1,l_2,\cdots,l_d)/n^{1/d}$-simulate mesh S, and this is optimal.

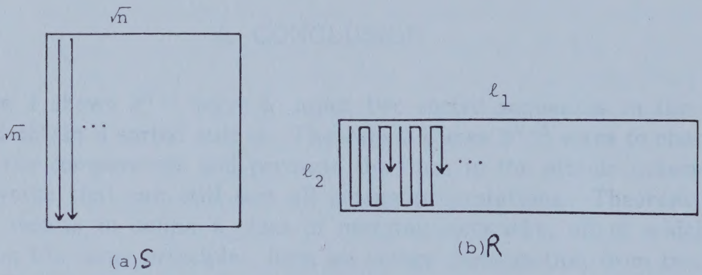
Corollary 1.1. An n-processor $(d-k)$-dimensional square mesh can $n^{k/d(d-k)}$-simulate an n-processor d-dimensional square mesh.

Theorem 2. Mesh S can 1-simulate mesh R, and this is optimal.

Sketch of Proof of Theorem 1:

Basis of induction: $d=2$. Without loss of generality, assume that $l_1\geq\sqrt{n}\geq l_2$. Think of S as consisting of $\sqrt{n}$ adjacent columns of length $\sqrt{n}$ each (see Fig. 1a).

Figure 1.



Now, snake these columns one after the other in R, as depicted in Fig. 1b. Note that each such snaked column occupies a width of $\sqrt{n}/l_2$ $(=l_1/\sqrt{n})$ along the l_1 direction of R. A data movement step between adjacent processors in the same column of S can clearly be simulated in $O(1)$ steps on R. It is trivial to design a data movement taking $O(l_1/\sqrt{n})$ on R and which simulates a data movement step between adjacent processors on the same row of S.

Inductive step: Without loss of generality, assume that $l_1\geq l_2\geq\cdots\geq l_d$. Note that $l_1\geq n^{1/d}\geq l_d$. Now, divide R along its first dimension into $n^{1/d}$ consecutive submeshes (which we call *R-chunks*) each of which is an $(l_1/n^{1/d})\times l_2\times\cdots\times l_d$ mesh of $n^{1-1/d}$ processors. Similarly divide S along (say) its first dimension into $n^{1/d}$ submeshes (which we call *S-chunks*) each of which is a $1\times n^{1/d}\times\cdots\times n^{1/d}$ mesh of $n^{1-1/d}$ processors. From here on we ignore the first dimension (of length 1) of an S-chunk, i.e. we consider it to be a $(d-1)$-dimensional square mesh of $n^{1-1/d}$ processors. Now, use each R-chunk to simulate an S-chunk. Of course, for this simulation, the S-chunks are assigned to the R-chunks in consecutive order. Before continuing with how an R-chunk simulates an S-chunk, let us pause to observe that two processors of S that are neighbours along S's first dimension are in two consecutive S-chunks, and that one data movement step in S between such processors can be simulated in R by a data movement taking time O(width of an R-chunk along its first dimension), i.e. $O(l_1/n^{1/d})$. We still need to show that a data movement step of S along its second, or third, ... , or dth dimension, can also be simulated by $O(l_1/n^{1/d})$ steps of R. Since each such data movement is between processors in the same S-chunk, it suffices to show that an R-chunk can $l_1/n^{1/d}$-simulate an S-chunk. We cannot yet use the induction hypothesis because S-chunks are $(d-1)$-dimensional while R-chunks are d-dimensional. Now, let C be an $(l_d.l_1/n^{1/d})\times l_2\times\cdots\times l_{d-1}$ mesh. It is not hard to show that an R-chunk can 1-simulate C (see [A] for details). Therefore it suffices to show that C can $l_1/n^{1/d}$-simulate an S-chunk. But C and an S-chunk satisfy the induction hypothesis, and therefore C can $\max(l_1.l_d/n^{2/d}, l_2/n^{1/d})$-simulate an S-chunk. If $l_2/n^{1/d}\geq l_1.l_d/n^{2/d}$ then C can $l_2/n^{1/d}$-simulate an S-chunk, and since $l_1\geq l_2$ it follows that C can $l_1/n^{1/d}$-simulate an S-chunk. If, on the other hand, $l_2/n^{1/d}<l_1.l_d/n^{2/d}$, then C can $l_1.l_d/n^{2/d}$-simulate an S-chunk. Since $l_d/n^{1/d}\leq 1$, this implies that C can $l_1/n^{1/d}$-simulate an S-

---

[†] This work was supported by the Office of Naval Research under Contract N00014-84-K-0502.

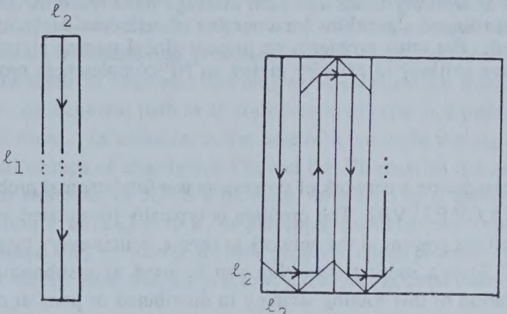
chunk. □

Sketch of Proof of Theorem 2:

The proof is by induction on d. We assume, without loss of generality, that $l_1 \geq l_2 \geq \cdots \geq l_d$.

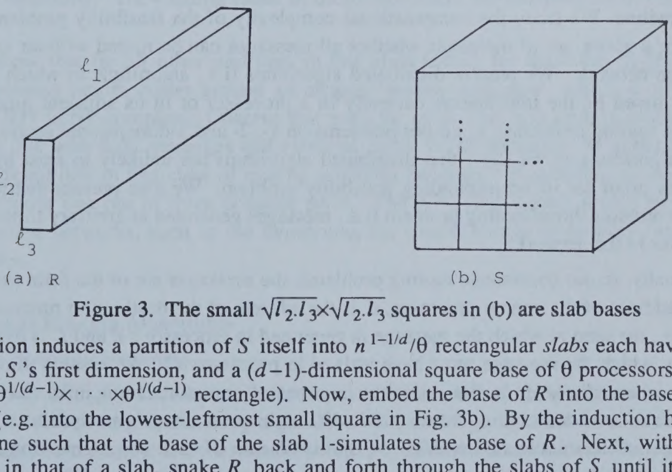
Basis: $d=2$. Embed R in S as follows. Partition S into $\sqrt{n}/l_2$ $(=l_1/\sqrt{n})$ rectangular slabs of dimensions $l_2 \times \sqrt{n}$ each, and then snake R through these slabs in the manner depicted in Fig. 2. Note that because of the way R moves from one slab of S to the next, some of the processors of S are idle (those in the empty triangular regions), while other processors of S are each simulating two of R's processors. It is obvious that this embedding enables S to simulate one step of R with $O(1)$ of its own steps.

Figure 2.



(a) R (b) S

Inductive Step: Think of l_1 as being the *depth* of R, and $l_2 \times \cdots \times l_d$ as being its *base*. Let θ be the number of processors of R's base, i.e. $\theta = l_2 . l_3 . \cdots . l_d$ $(=n/l_1)$. Observe that $\theta \leq n^{1-1/d}$. Partition the $(d-1)$-dimensional, $n^{1-1/d}$-processor square determined by the last $d-1$ dimensions of S into $n^{1-1/d}/\theta$ $(d-1)$-dimensional squares of θ processors each. Fig. 3 illustrates for the case $d=3$.



(a) R (b) S

Figure 3. The small $\sqrt{l_2 . l_3} \times \sqrt{l_2 . l_3}$ squares in (b) are slab bases

This partition induces a partition of S itself into $n^{1-1/d}/\theta$ rectangular *slabs* each having a depth of $n^{1/d}$ along S's first dimension, and a $(d-1)$-dimensional square base of θ processors (i.e. each slab is a $n^{1/d} \times \theta^{1/(d-1)} \times \cdots \times \theta^{1/(d-1)}$ rectangle). Now, embed the base of R into the base of a "corner" slab in S (e.g. into the lowest-leftmost small square in Fig. 3b). By the induction hypothesis, this can be done such that the base of the slab 1-simulates the base of R. Next, with its own base embedded in that of a slab, snake R back and forth through the slabs of S until it is completely embedded in S. We do not elaborate on this since it is an obvious generalization of the snaking done in Fig. 2b. However there is one point worth mentioning about the above embedding process: It is crucial that the depth of a slab $(=n^{1/d})$ is no smaller than the length of a base's side $(=\theta^{1/(d-1)})$ in order for R to shift smoothly from one slab to another (this condition was satisfied in Fig. 2b since we had $l_2 \leq \sqrt{n}$). A data movement in R along its first dimension can obviously be simulated in $O(1)$ time in S. A data movement along any of the remaining $d-1$ dimensions of R can also be simulated in $O(1)$ time in S, because (by the induction hypothesis) the base of a slab can 1-simulate the base of R. □

References

[A] M.J. Atallah, "On Multidimensional Arrays of Processors," Purdue CS Tech. Rept. 528

Continuous Routing and Broadcasting on Interconnection Networks

*Yukon Chang**
Computer Science Dept.
Pennsylvania State University
University Park, PA 16802

Susanne Hambrusch **
Dept. of Computer Sciences
Purdue University
West Lafayette, IN 47907

Janos Simon *
Dept. of Computer Science
University of Chicago
Chicago, IL 60637

Abstract. We study the problem of routing and broadcasting messages in a network in which the messages are generated at arbitrary times and each message must reach its destination by a specific deadline. We present distributed algorithms for a number of such continuous routing problems on the linear array and the mesh. For other problems we present global routing algorithms or we show that distributed algorithms are unlikely to exist by giving an NP-completeness proof for their feasibility problem.

1. Introduction

The routing of messages on a network of processors is a fundamental problem in distributed and parallel environments [A,GP,P,U,VB]. The problem is typically formulated as finding an algorithm that, given a set of messages present in the network at time t, routes every message to its destination as quickly as possible. Such a routing algorithm can be used as a subroutine to simulate shared memory. Direct application of this routing strategy in distributed or parallel computation, however, may not be efficient: computation cannot continue until all the messages have been routed, even though different messages arrive at their destinations at different times and processors could proceed with local computation.

In this paper we consider a more general formulation, the *continuous routing problem*, in which messages are generated at processors at arbitrary times and each message must reach its destination by a specific deadline. We study the computational complexity of the feasibility problem, in which we determine for a given set of messages whether all messages can be routed without violating the constraints of the network. We present distributed algorithms (i.e., algorithms in which the routing strategy is only based on the information currently in a processor or in its adjacent processors) for some continuous routing problems. For other problems on 1-, 2- and 3-dimensional arrays we present global routing algorithms or we show that distributed algorithms are unlikely to exist by giving an NP-completeness proof for its corresponding feasibility problem. We also present distributed algorithms for the continuous broadcasting problem (i.e., messages generated at arbitrary times are routed to every processor in the network).

More formally, in our continuous routing problems, the messages are of the form $m=(s,d,t,f)$, where s is the address of the source processor, d is the address of the destination processor, t is the start-up time (i.e., the time at which the message is generated in processor s), and f is the finish time (i.e., the time by which the message must have arrived in processor d). We assume throughout that $f-t \geq dist(s,d)$, where $dist(s,d)$ is the minimum number of communication links that have to be traversed in order to send message m from s to d. We assume we are given a synchronous network which has two parameters associated with it: b, the *bandwidth* of each communication link, and c, the internal *storage capacity* of each processor. At each time step a processor (PE) can send at most b messages along each one of its links, and at each time step at most c messages can be stored in a PE. The solution of a global routing algorithm consists of the description of the routing path from the source to the destination PE for every message and the time at which each communication link is used. A solution is feasible if it observes both the bandwidth and storage capacity restrictions.

---

* This work was supported in part by USARO Grant DAAG29-82-K-0110 and in part by NSF Grant DCR-8407256.

** This work was supported in part by the ONR Contract N00014-84-K-0502 and NSF Grant DMC-8413496.

We say that a routing problem is *tight* if every message $m=(s,d,t,f)$ satisfies $f-t=dist(s,d)$. In some cases, tight routing problems can be solved more efficiently that general, non-tight ones. We present characterizations for testing feasibility of tight broadcasting in α-dimensional arrays, $\alpha \geq 1$, and other interconnection structures, and of tight top-bottom routing on the mesh (i.e., the destination PEs are in the first row and the destination PEs are in the last row). Both characterizations lead to distributed algorithms.

We consider a class of continuous routing problems on 1-, 2-, and 3-dimensional arrays - also known as linear array, mesh, and cube - to which we give mnemonic names drawn from the set $\{1D,2D,3D\} \times \{T,\varepsilon\} \times \{E,\varepsilon\}$, where ε is the empty string. The meaning of the mnemonics is as follows: αD indicates that an α-dimensional processor array is used. The letter 'T' indicates that the all messages are *tight*. The letter 'E' indicates that only *extremal paths* are allowed in the routing which are defined as follows. An extremal path in an α-dimensional array is a path that changes 'direction' a minimum number of times. In addition, in the case of a non-tight message, the message can only be stored in the internal storage of exactly one PE, and this PE must be one at which the routing path changes direction. For example, in 2DTE a message from $s=(x_1,y_1)$ to $d=(x_2,y_2)$ either takes the lower extremal path from s to (x_2,y_1) to d, or the upper extremal path from s to (x_1,y_2) to d. In 2DE, the message is stored for $f-t-dist(s,d)$ time pulses in either processor s, d, or in (x_2,y_1) if the lower extremal path is taken. Note that for practical purposes extremal paths are desirable since they have a short description.

On the linear array every tight message has only one routing path available, which is also the extremal path, and thus 1DT and 1DTE are the same problem. Let k be the number of messages to be routed during the entire computation. We present an $O(k \log k)$ time algorithm for testing feasibility for 1DT. For 1DE with $b=1$ and $c=1$ and for 2DTE with $b=1$ we present an $O(k^2)$ time algorithm for testing feasibility. The running times of these algorithms are independent of the size of the network.

We show that for all other problems in this class testing for feasibility is NP-complete. The NP-completeness proofs center around an elegant 'master reduction', which is first introduced to show that 2DT is NP-complete. Observe that even the most restricted routing problem on the cube, namely the one with tight messages and extremal paths, is NP-complete. In general, it appears that testing for feasibility in this class of routing problems becomes NP-complete as soon as every message can choose one out of three possible paths. The NP-completeness proofs generalize to other interconnection networks, such as the hypercube, the shuffle-exchange network, and the cube connected cycles.

2. Continuous Routing Algorithms

2.1 Global Routing Algorithms

Let $m_i=(s_i,d_i,t_i,f_i)$, $1 \leq i \leq k$, be the messages to be routed during the entire computation. In this section we describe two global routing algorithms. The first algorithm solves continuous routing of k tight messages on the linear array in time $O(k \log k)$ for arbitrary bandwidth. The second one solves, in time $O(k^2)$, extremal paths routing of k tight messages on the mesh (2DTE) with $b=1$ and also extremal paths routing on the linear array (1DE) with $b=1$ and $c=1$. In Section 4 we show that both problems are NP-complete for other values of b and c.

Before describing the algorithms, we give the definition of a *timed path*. A timed path for message $m=(s,d,t,f)$ is the sequence $[(x_0,t), (x_1,t+1), \dots, (x_{f-t-1}, f-1), (x_{f-t}, f)]$, where $x_0=s$ and $x_{f-t}=d$, and where for consecutive entries (x,z) and $(y,z+1)$ either PE_x and PE_y are directly connected by a communication link, or $x=y$. The timed path represents the *transitions* message m makes. If $x \neq y$, the action associated with the transition $[(x,z),(y,z+1)]$ is to send message m from PE_x to PE_y between time z and $z+1$. When $x=y$, the action associated is to store message m in PE_x between time z and $z+1$.

Every processor in a network of bandwidth b and storage capacity c has two types of storage: I/O registers and storage locations. The *I/O registers* are used exclusively for storing messages immediately before or after transmission (to or from adjacent PE's) and their number is proportional to b. Each one of the c *storage locations* of a PE consists of enough registers to store one message. These assumptions are similar to the actual implementation of the CHiP architecture [S]. Note that tight messages do not use any storage locations, only I/O registers. A solution to a routing problem thus satisfies the storage and bandwidth restrictions if for every time instance z

$| \{ m_i \mid [(x,z), (y,z+1)]$ is a transition of $m_i \} | \leq b$ for every link (x,y), and
$| \{ m_i \mid [(x,z), (x,z+1)]$ is a transition of $m_i \} | \leq c$ for every PE_x.

Note that the link (x,y) is viewed as consisting of two directed links, each one of bandwidth b.

We now describe the algorithm for 1DT. We are given a linear array of size n and k tight messages. In the following, we show how to determine in $O(k \log k)$ time the minimum bandwidth $b_{\min}$ needed in order to achieve a feasible routing. Since the messages are tight, the paths of right-bound messages (i.e., the ones with $s_i<d_i$) do not interfere with the paths of left-bound messages (i.e., the ones with $s_i>d_i$), and we thus consider them independently. Assume we have k_1 right-bound messages, and w.l.o.g. let them be $m_1, \cdots, m_{k_1}$. The timed path for m_i is $[(s_i, t_i), (s_i+1, t_i+1), \ldots, (d_i, t_i+d_i-s_i)]$. Two right-bound messages m_i and m_j use the communication link from u to $u+1$, $1 \leq u<n$, at the same time if and only if $s_i \leq u<u+1 \leq d_i$, $s_j \leq u<u+1 \leq d_j$, and $t_i-s_i=t_j-s_j$. We thus partition the k_1 messages into sets of messages with the same t_i-s_i value. Within each set we view every message as an interval (s_i,d_i). The number of messages using the communication link from PE_u to PE_{u+1} at the same time is equal to the number of intervals (s_i,d_i) in the set with $s_i \leq u<u+1 \leq t_i$. Thus b_r (resp. b_l), the minimum bandwidth needed by the right-bound (resp. left-bound) messages, can be computed in $O(k_1 \log k_1)$ (resp. $O((n-k_1)\log(n-k_1))$ time, and $b_{\min} = \max(b_r, b_l)$.

Theorem 2.1. The feasibility problem for 1DT and 1DTE on a linear array of bandwidth b can be solved in $O(k \log k)$ time.

Proof. Determine $b_{\min}$ using the algorithm described above and compare it to b. ☐

Next we describe the algorithm that solves 2DTE in time $O(k^2)$ when $b=1$. Recall that the value of c is irrelevant in the case of tight messages. Assume the mesh is of size $m \times n$. Let position (0,0) be the bottom leftmost corner, and (c,r) denote the position of column c and row r. Let $m = ((w, x), (y, z), t)$ be a message (we omit the finish time entry for tight messages). If $w=y$ (resp. $x=z$), only one path, the *column path* (resp. *row path*), is available for m. If $w \neq y$ and $x \neq z$, two extremal paths, the *upper (extremal)* and the *lower (extremal) path* are available. Message m takes the upper (resp. lower) path, if m moves first into the correct row (resp. column). Detecting interference between two paths when $b=1$ can easily be done in constant time. If a column (resp. row) path interferes with another column (resp. row) path, no feasible routing exist. If a column or a row path interferes with the upper (resp. lower) path of message m, then message m must take the lower (resp. upper) path. In addition, every message must choose exactly one path.

Based on these observations, we form a boolean formula in CNF in which every clause contains 2 variables (2SAT) so that the formula is satisfiable if and only if a feasible routing exists. The formula consists of $O(k^2)$ clauses and at most $2k$ variables. Recall that 2SAT can be solved in time proportional to the number of clauses in the formula. For every message m_i with both source and destination in the same column (resp. row), we create a column (resp. row) variable v_i^C (resp. v_i^R). For every message m_j with source and destination in different columns and rows, we create two variables, an upper variable v_j^U and a lower variable v_j^L.

Next we determine, in $O(k^2)$ time, which extremal paths interfere with each other. For every column (resp. row) variable v_i^C (resp. v_i^R) we add the clause (v_i^C **or** false) (resp. (v_i^R **or** false)). For every message m_i with source and destination in different columns and rows, we add the clauses (v_i^U **or** v_i^L) and ($\overline{v_i^U}$ **or** $\overline{v_i^L}$), where $\overline{v}$ stands for the negation of v. Finally, we need to encode the interference of message paths into the formula. If the upper (resp. lower) path of message i interferes with the column or the row path of another message, we add the clause (v_i^L **or** false) (resp. (v_i^R **or** false)). If the two upper paths, the two lower path, or the upper and the lower path of message m_i and message m_j interfere, we add, respectively, the clauses (v_i^L **or** v_j^L), (v_i^U **or** v_j^U), or (v_i^L **or** v_j^U). It is straightforward to show that the so formed boolean formula is satisfiable if and only if a feasible routing exists. We thus have the following result:

Theorem 2.2. The feasibility problem for 2DTE with $b=1$ can be solved in time $O(k^2)$.

By a similar algorithm (whose details can be found in [CHS]) we obtain the following result:

Theorem 2.3. The feasibility problem for 1DE with $b=c=1$ can be solved in $O(k^2)$ time.

To conclude this section, we observe that the idea used in the previous two algorithms can be applied to continuous routing problems on arbitrary networks in which every message has at most two paths available, and both storage capacity and bandwidth are one. The running time depends now on the time needed to detect interference between two paths. We state this generalization in the following corollary.

Corollary 2.4. Let N be an arbitrary network with $b=1$ and $c=1$, and assume k messages are to be routed. If every message must take one out of two timed paths, the feasibility problem can be solved in $O(Tk^2)$ time, where T is the time required to detect interference between two timed paths.

2.2 *A Distributed Algorithm for Top-Bottom Routing*

While the feasibility problem of many routing problems becomes NP-complete as soon as every message can choose one out of three paths, another interesting routing problem, namely top-bottom routing on a mesh, can be solved by a distributed algorithm. In *top-bottom routing* all the source PEs are located in the top row of the mesh (i.e., row $n-1$), and all the destination PEs are located in the bottom row of the mesh (i.e., row 0). The PEs in between thus serve as 'switches'. We have the following result:

Theorem 2.5 Continuous tight top-bottom routing on an $m \times n$, $m \leq n$, mesh with unit horizontal bandwidth and vertical bandwidth 2, is feasible if (i) no two messages are generated at the same source PE at the same time, and (ii) no two messages arrive at the same destination PE at the same time.

Proof. If (s_i, d_i, t_i) is a message, then s_i (resp. d_i) is the column on row $n-1$ (resp. 0) of the source (resp. destination) PE. We say the a message is right-bound (resp. left-bound) if $s_i \leq d_i$ (resp. $s_i > d_i$). The algorithm routes the right-bound and the left-bound messages independently. (This can be done since the vertical bandwidth is 2 and the horizontal links are used by right- and left-bound messages in different directions.) Messages are first sent to the column containing the destination PE. If two right-bound (resp. left-bound) messages want to use the same horizontal communication link at the same time, the message with the destination in a closer column uses the vertical link.

More formally, the algorithm for right-bound messages works as follows. At any time t, a PE at an arbitrary position (c, r) contains at most 2 right-bound messages which are stored in its I/O registers. If there is only one message stored in the PE at (c, r), say a message with destination d_i, then this message is sent from (c, r) to $(c+1, r)$ between time t and $t+1$ if $d_i > c$, and it is sent from (c, r) to $(c, r+1)$ otherwise. If the PE at position (c, r) contains two messages, say with destinations d_i and d_j and $d_i < d_j$, then the message with destination d_i is sent along the link from (c, r) to $(c, r+1)$ and the message with destination d_j is sent from (c, r) to $(c+1, r)$ between time t and $t+1$. After this step every PE contains again at most 2 right-bound messages. The proof that every PE in row 0 has never 2 right-bound messages stored in it is straightforward and is omitted. ☐

3. Continuous Broadcasting Algorithms

In this section we study another elementary problem arising in a parallel and distributed environment: that of *broadcasting* messages from source PEs to all other PEs in the network. Each such broadcast message has a source s and a start-up time t, and is denoted by $m=(s,t)$. We consider the case of tight broadcasting in which message m has to be routed so that it arrives at every processor w at time $t+dist(s,w)$, where $dist(s,w)$ is the shortest distance from s to w in the network. Recall that in this case the internal storage capacity of each processor is irrelevant. The feasibility problem in broadcasting is analogous to the one in routing: Given a set of messages and a network with bandwidth b, determine whether all messages can be broadcasted without violating the bandwidth constraint. We start with defining the TAGGED k-BIN PACKING problem and give a lemma that characterizes the feasible instances for this problem. This lemma plays a central role in the characterization for testing feasibility for broadcasting on processor arrays and other types of networks. Distributed broadcasting algorithms follow from this characterization. In this abstract we only present the algorithm for the mesh, and we refer to [CHS] for generalizations.

3.1. TAGGED k-BIN PACKING Problem

In the TAGGED k-BIN PACKING problem we are given k bins, referred to as bin 0, ..., bin $k-1$ with capacities $b_0, \ldots, b_{k-1}$, respectively, and a finite set U of items with unit size. Each item $x \in U$ has a *tag* T_x associated with it, $T_x \subseteq \{0,1,\cdots,k-1\} = [\mathbf{k}]$. Tag T_x specifies in which bins item x can be placed; i.e., if $i \in T_x$, then x can be placed into bin i. Let a_T be the number of items with tag T, $T \subseteq [\mathbf{k}]$. For every set S, $S \subseteq [\mathbf{k}]$, let $ITEM(S) = \sum_{T \subseteq S} a_T$ and $CAPACITY(S) = \sum_{i \in S} b_i$. $ITEM(S)$ is a lower bound on the number of items that must be placed into the bins in set S, and $CAPACITY(S)$ is the maximum number of items that can be placed into the bins in set S. Let $SLACK(S) = CAPACITY(S) - ITEM(S)$. The proof of the next lemma can be found in [CHS].

Lemma 3.1 The TAGGED k-BIN PACKING problem has a feasible solution if and only if $SLACK(S) \geq 0$ for every subset S of $[\mathbf{k}]$.

3.2 Broadcasting on the Mesh

In this section we describe a distributed broadcasting algorithm for the mesh. The algorithm succeeds in broadcasting all the messages if a feasible solution exists. For every PE $u = (c_u, r_u)$ in the mesh, denote its four neighbors (c_u+1, r_u), (c_u, r_u+1), (c_u-1, r_u), and (c_u, r_u-1), if they exist, as u_0, u_1, u_2, and u_3, respectively. Define for PE u up to 8 disjoint regions as follows:

$$R_0(u) = \{(c,r) \mid c > c_u \text{ and } r = r_u\}, \quad R_{0,1}(u) = \{(c,r) \mid c > c_u \text{ and } r > r_u\},$$
$$R_1(u) = \{(c,r) \mid c = c_u \text{ and } r > r_u\}, \quad R_{1,2}(u) = \{(c,r) \mid c < c_u \text{ and } r > r_u\},$$
$$R_2(u) = \{(c,r) \mid c < c_u \text{ and } r = r_u\}, \quad R_{2,3}(u) = \{(c,r) \mid c < c_u \text{ and } r < r_u\},$$
$$R_3(u) = \{(c,r) \mid c = c_u \text{ and } r < r_u\}, \quad R_{3,0}(u) = \{(c,r) \mid c > c_u \text{ and } r < r_u\}.$$

As shown in Figure 3.1, the subscripts 0, 1, 2, and 3 encode the directions east, north, west, and south, respectively. Regions $R_0(u)$, $R_1(u)$, $R_2(u)$, and $R_3(u)$ are the axes with u as the origin, while regions $R_{0,1}(u)$, $R_{1,2}(u)$, $R_{2,3}(u)$, and $R_{3,0}(u)$ are the four quadrants.

Let t be some arbitrary time. All the messages originating from some PE s with a start-up time of $t-dist(s,u)$ arrive at PE u at the same time and thus compete for communication links. We refer to these messages as the *relevant* messages for u at time t. Let $a_i(u,t)$ (resp. $a_{i,i+1}(u,t)$) be the number of relevant messages for u at time t that originate in region $R_i(u)$ (resp. $R_{i,i+1}(u)$) of the mesh. Note that the addition in the subscripts is addition mod 4, and that, when there is no ambiguity as to what u and t are, the arguments (u) and (u,t) are dropped from R's and a's. We then have the following theorem.

Theorem 3.2. Tight continuous broadcasting on a mesh of bandwidth b is feasible if and only if the following conditions, denoted as (Eq. I), hold for all PEs u, all time instances t, and $i = 0,1,2,3$:

$$a_i \le b, \qquad a_i + a_{i,i+1} + a_{i+1} \le 2b,$$

$$a_i + a_{i,i+1} + a_{i+1} + a_{i+1,i+2} + a_{i+2} \le 3b, \qquad \text{and } \sum_{j=0}^{3} a_j + \sum_{j=0}^{3} a_{j,j+1} \le 4b.$$

Proof. The only if part is shown easily. For any u and t, the left hand side of each inequality in (Eq. I) represents the number of messages that must use one of a certain set of links to reach u, while the right hand side is the maximum number of messages these links can accommodate collectively at any given time. To show the if part, we present a broadcasting algorithm and show that, if broadcasting is successful up to time t and (Eq. I) is true for all u at time $t+1$, then the algorithm will carry out the broadcasting successfully one more time step to time $t+1$. This serves as the inductive step for the proposition that, if (Eq. I) is true for all u and t, then broadcasting is feasible. To establish the inductive basis, we observe that for $t=0$ (Eq. I) is true because all a's are 0.

Now assume that broadcasting has been done successfully up to time t. We must show that by the end of time $t+1$ every PE has received all the messages relevant for it at time $t+1$ and that the bandwidth restriction is observed. Consider an arbitrary PE u at time $t+1$. At time t, all the relevant messages for u at time $t+1$ reside in u's neighboring PEs. More precisely, a relevant message originating from region $R_i(u)$ resides in u_i, and a relevant message originating from region $R_{i,i+1}(u)$ has one copy of it in PE u_i and one in PE u_{i+1}. In the first case the message must use the link $u_i \rightarrow u$. In the second case, however, only one copy of the message needs to be sent to PE u, and either the link $u_i \rightarrow u$ or the link $u_{i+1} \rightarrow u$ has to be used.

The algorithm works as follows. At time $t+1$, PE u first acquires the values of $a_i(u,t+1)$ and $a_{i,i+1}(u,t+1)$ from its neighbors. These eight values are easily determined (by u's neighbors) from $a_i(u_i,t)$, $a_{i-1}(u_i,t)$, $a_{i+1}(u_i,t)$, $a_{i-1,i}(u_i,t)$, and $a_{i,i+1}(u_i,t)$, $i=0,1,2,3$. Note that, if processor u_i contains a message $\hat{m}=(\hat{s},\hat{f})$ it can determine whether or not message $\hat{m}$ is relevant for PE u at time $t+1$. In order for processor u to decide which messages to receive from whom, u solves a special case of the TAGGED k-BIN PACKING problem. In this problem $k=4$, the bin capacities equal the bandwidth b, and the items and tags are as follows. For every message relevant for u at time $t+1$ that originated from $R_i(u)$ it creates an item with tag $\{i\}$, and for every message relevant at time $t+1$ that originated from $R_{i,i+1}(u)$ it creates an item with tag $\{i,i+1\}$. It is easy to see that (Eq. I) is equivalent to the condition that $SLACK(S) \ge 0$ for every subset S of $\{0,1,2,3\}$. Since the 4-BIN PACKING problem can be solved in $O(b)$ time, PE u makes the decisions also in $O(b)$ time. If an item is assigned to bin i, then the corresponding message is requested from processor u_i and it uses the link $u_i \rightarrow u$. Finally, those messages with start-up time $t+1$ and source u will be in u at time $t+1$. This completes the proof of Theorem 3.2. □

The above algorithm is not a fully distributed algorithm as defined in [U], because each PE's decision depends not only on information stored in the PE but also on that stored in its neighbors. A fully distributed algorithm, however, does not exist in continuous broadcasting. For such an example see [CHS].

4. NP-Completeness Results

In this section we give the NP-completeness proofs of the routing problems claimed to be NP-complete in Section 1. If not otherwise stated, we assume that both the bandwidth b of the network and the storage capacity c of every processor are equal to 1, but our results hold as long as b and c are polynomial in the size of the network. We start with 2DT, in which we are given a 2-dimensional mesh of size $m \times n$ and k tight messages $m_i = (s_i,d_i,t_i)$, $1 \le i \le k$. Recall that in Section 2.1 we gave an in $O(k^2)$ time algorithm for the feasibility problem on the mesh when every message must take one of the (at most) two extremal paths. We now show that, if the message paths have 'more freedom', the problem of deciding whether or not a feasible routing exists becomes NP-complete. We do so by reducing monotone 3SAT to 2DT. In monotone 3SAT every clause of a boolean formula in conjunctive normal form contains either 3 negated or 3 unnegated variables [GJ]. This reduction is, in a sense, the 'master reduction' of this section, since the other NP-completeness proofs are obtained either by restriction or by modifications of it.

Theorem 4.1. 2DT is NP-complete.

Proof. Let $C_0, C_1, \cdots, C_{k-1}$ be the k clauses of the boolean formula. Let $x_0, x_1, ..., x_{l-1}$ be the l variables used in the k clauses. We call a clause $C_u = [z_{u,0}$ **or** $z_{u,1}$ **or** $z_{u,2}]$, a positive clause if $z_{u,i} \in \{x_0, x_1, ..., x_{l-1}\}$, and a negative clause if $z_{u,i} \in \{x_0, x_1, ..., x_{l-1}\}$, i=0,1,2. Assume that the three variables in every clause are distinct. Our transformation of an instance of monotone 3SAT into one of 2DT uses a mesh of size $14k \times 14k$ and generates $O(k)$ messages. We again denote the bottom leftmost corner of the mesh as position (0,0), and the PE in column c and in row r is said to be at position (c, r). The following two properties hold for every message $m = (s, d, t)$ generated in the transformation:

1) If the source (resp. destination) PE is in column c_s (resp. c_d) and in row r_s (resp. r_d), then $c_s \le c_d$ and $r_s \le r_d$.

2) The start-up time t of the message m is $c_s + r_s$.

Thus the start-up time of a message is determined by the position of the source PE, and we omit the start-up time entry and denote a message by $((c_s, r_s), (c_d, r_d))$. Observe that, because of property 2, if a message arrives at processor (x, y), it does so at time $x+y$.

With every positive (resp. negative) clause C_u we associate a positive (resp. negative) processor block B_u of size 14×14 along the diagonal of the mesh. Let $p_u = 14u$ and $q_u = 14u+13$. Then, B_u has processors (p_u, p_u), (p_u, q_u), (q_u, q_u), and (q_u, p_u) at its four corners. The messages generated in the transformation are classified into 5 groups:

Variable Messages. In block B_u create three variable messages (one for every variable $z_{u,i}$ in C_u), namely $((p_u+2i, p_u+2i), (q_u-2i, q_u-2i))$, i=0,1,2.

Control Messages. In block B_u create three control messages, namely $((p_u+6, p_u+2i+1), (p_u+7, p_u+2i+1))$, i=0,1,2, if B_u is a positive block and $((q_u-7, q_u-2i-1), (q_u-6, q_u-2i-1))$, i=0,1,2, if B_u is a negative block.

Enforcer Message. In block B_u create one enforcer message. It is $((p_u+6, p_u), (p_u+7, p_u+4))$ for a positive block, and $((q_u-7, q_u-4), (q_u-6, q_u))$ for a negative block.

Consistency Messages. If $z_{u,i}$ and $z_{v,j}$ are two adjacent occurrences of a variable (i.e., clauses $C_{u+1}, \cdots, C_{v-1}$ do not contain this variable), then create one main consistency messages, namely, $((q_u-2i-1, q_u-2i-1), (p_v+2j+1, p_v+2j+1))$, and two auxiliary consistency messages, $((q_u-2i-1, q_u-2i-1), (q_u-2i, q_u-2i))$ and $((p_v+2j, p_v+2j), (p_v+2j+1, p_v+2j+1))$.

Divider Messages. Two identical messages $((i, i)\ (i+1, i+1))$ for every i, $i \in \{0, 2, \cdots, 14k\}$.

The divider messages make every PE on the diagonal at an even position the source of two messages, and every PE on the diagonal at an odd position the destination of two messages. This forces the variable and consistency messages (which have both source and destination PE on the diagonal) to select a path that either stays entirely below the diagonal (i.e., a lower path), or one entirely above the diagonal (i.e., an upper path). If the first move of a variable message $((w, w), (x, x))$ is to position $(w, w+1)$ (resp. $(w+1, w)$), then this variable message must take an upper (resp. lower) path and arrive at (x, x) from position $(x-1, x)$ (resp. $(x, x-1)$). In positive blocks the enforcer message controls the number of variable messages that can cross from column p_u+6 to p_u+7 below the diagonal. See Figure 4.1 for an example of a positive block.

Showing that the routing instance constructed above has a feasible routing if the boolean formula is satisfiable is left to the reader. We proceed proving the other direction. Assume that the 2DT instance has a feasible routing. From this routing we show how to obtain a variable assignment that satisfies the boolean formula. Recall that in our construction every variable and every consistency message takes either an upper or a lower path. We claim that the variable messages corresponding to the same variable either all take upper paths or all take lower paths. Let $z_{u,i}$ and $z_{v,j}$ be two consecutive occurrences of a variable x. Suppose that in B_u the i-th variable message takes an upper path. As described earlier, this variable message arrives at its destination (q_u-2i, q_u-2i) from (q_u-2i-1, q_u-2i). This forces the auxiliary consistency message $((q_u-2i-1, q_u-2i-1), (q_u-2i, q_u-2i))$ to take a lower path. Thus the main consistency message $((q_u-2i-1, q_u-2i-1), (p_v+2j+1, p_v+2j+1))$ has to take an upper path, which then forces the auxiliary consistency message $((p_v+2i, p_v+2i), (p_v+2i+1, p_v+2i+1))$ in block B_v to take a lower path and it forces the variable message corresponding to x in block B_v to take an upper path. The assignment of the variables in the boolean formula is obtained as follows. Assign a variable the value true (resp. false) if the corresponding variable messages take upper (resp. lower) paths. It is easy to see that this assignment indeed satisfies the boolean formula, and Theorem 4.1 follows. □

Since the routing instance constructed for 2DT in the previous theorem can also be considered as an instance of 2D, 3DT and 3D, the NP-completeness of these problems follows trivially. Observe that among all the messages generated in the previous proof only the enforcer message requires three choices for the path. It is thus not surprising that 3DTE (i.e., routing tight messages using extremal paths on the cube) is also NP-complete, since there are six extremal paths for every tight message in the cube. We next show that 1D, the general continuous routing problem on the linear array is NP-complete. Recall that in 1D we are given a linear array of n PE's and k messages $m_i = (s_i, d_i, t_i, f_i)$, $1 \le i \le k$.

Theorem 4.2. 1D is NP-complete.

Proof. We first show that the version of 1D in which every message m_i has $s_i \le d_i$, and in which no communication link from PE_x to PE_{x-1} is allowed to be used is NP-complete. The modification for the general case is left to the reader. If only the communication links from PE_x to PE_{x+1} can be used, we can interpret the instance of 2DT constructed from a boolean formula in Theorem 4.1 as an instance of 1D as follows. Since the paths of messages move either up or to the right, the sum of the two coordinates of a PE containing a message in the mesh at time t differs from the sum of the coordinates of the PE containing this message at time $t+1$ by 1. We now interpret the first coordinate (i.e., the column) as the address of a PE in the linear array, and the sum of the two coordinates in the mesh as the time in the linear array. A move of a message from position (x, y) to position $(x+1, y)$ in the mesh is interpreted as a move from PE_x to PE_{x+1} between time $x+y$ and $x+y+1$ in the linear array. A move of a message from position (x, y) to position $(x, y+1)$ in the mesh is interpreted as keeping the message in PE_x between time $x+y$ and $x+y+1$ in the linear array. For every message $((x_1, y_1), (x_2, y_2))$ created for the mesh in the transformation, we create the message $(x_1, y_1, x_1+y_1, x_2+y_2)$ on the linear array. Hence, the given instance of monotone 3SAT is satisfiable if and only if the so created instance on the linear array has a feasible routing. □

Theorem 4.3. 2DE, 3DE and 3DTE are NP-complete.

The proof for 3DTE is an interesting generalization of the one of Theorem 4.1 to 3 dimensions. We refer to [CHS] for details.

In Section 2 we showed that the feasibility problem of 2DTE with $b=1$ and the one for 1DE with $b=1$ and $c=1$ can be solved in $O(k^2)$ time. We now show that, if more bandwidth and storage locations are available, both feasibility problems become NP-complete. Recall that in the master transformation all messages, except the enforcer messages, are able to take an extremal path. In a positive block, the enforcer message has to take a non-extremal path if the first and the third variable message take the lower and the second variable message takes the upper extremal path. Consider 2DTE with $b=2$. The 'master' transformation is then modified as follows. In every block, we add control messages which make the bandwidth of all, except one, communication link equal to 1. In a positive block, the communication link from (p_u+6, p_u+4) to (p_u+7, p_u+4) does not get such a control message. Also add one new message, namely $((p_u+6, p_u+2), (p_u+7, p_u+4))$. When only the middle link is open (i.e., only the second variable message takes the upper path), the new message takes the lower path and this leaves an extremal path to the enforcer message. We leave the details to the interested reader.

Theorem 4.4. 2DTE with $b=2$, and 1DE with either $b=2$ or $c=2$ are NP-complete.

One interesting feature of continuous routing is its similarity to wire routing problems. Karp et al. proved in [KL] that it is NP-complete to determine whether the optimal width of one-turn routing on the mesh is less than $\lceil n/2 \rceil$-2. The proof of Theorem 4.4 can be modified to achieve the following stronger result.

Corollary 4.5. It is NP-complete to determine whether the optimal width of one-turn routing is equal to 2.

We conclude this section by generalizing our NP-completeness results to other classes of graphs (which are considered to be networks). Intuitively, we view a simple path in a given graph as a linear array. If a simple path is "long enough", then the reduction from 3SAT to 1D can be used to show that continuous routing on the given graph is NP-complete. More precisely, let Γ denote a class of graphs. We say Γ is *dense* if there exists a polynomial p such that for every integer i there is a graph $G_i = (V_i, E_i)$ in Γ with $i \le |V_i| \le p(i)$. We say Γ has *long paths* if there exists a constant ε such that for every i, graph G_i in Γ contains a simple path π of length $|V_i|^{\varepsilon}$.

Theorem 4.6. Let Γ be a class of graphs. If

1) Γ is dense and has long paths, and

2) for every i, G_i and a simple path of length $|V_i|^{\varepsilon}$ can both be computed in polynomial time,

then continuous routing on Γ is NP-complete.

Proof. Given an instance of monotone 3SAT containing k clauses, let i be the integer such that $|V_i| = 14k^{1/\varepsilon}$. Using assumption 2, compute G_i and a long path π of length at least $14k$ in polynomial time. For every edge (u,v) in G_i such that u is on π and v is not, add $28k$ control messages $(u,v,t,t+1)$ and $(v,u,t,t+1)$, $t=0,1,\ldots,14k-1$. If we now view π as a linear array, then the construction of the proof of Theorem 4.2 gives the claimed result. ☐

Corollary 4.7. Continuous routing problems on the following networks are NP-complete: ring, hypercube, cube connected cycle, X-tree, and shuffle exchange.

5. References.

[A] R. Aleliunas, "Randomized Parallel Communication", *Proceedings of the 1st Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 1982, pp. 60-72.

[BH] A. Borodin and J. E. Hopcroft, "Routing, Merging and Sorting on Parallel Models of Computation", *Journal of Computer and System Sciences*, Vol. 30, Nr. 1, 1985, pp. 130-145.

[CHS] Y. Chang, S. E. Hambrusch, and J. Simon, "On the Complexity of Continuous Routing", Tech. Report CS-85-12, Pennsylvania State University, June 1985.

[GJ] M. Gary and D. Johnson, *Computers and Intractability*, 1979, pp. 259.

[GP] Z. Galil and W. J. Paul, "An Efficient General-Purpose Parallel Computer", *Journal of ACM.*, 30(2), April 1983, pp. 360-387.

[KL] R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. Vazirani, and V. Vazirani, "Global Wire Routing in Two-Dimensional Arrays," *roceedings of the 24th FOCS*, 1983, pp. 453-459.

[P] N. Pippenger, "Parallel Communication with Limited Buffers", *Proceedings of the 25th FOCS*, 1984, pp. 127-136.

[S] L. Snyder, "Overview of the CHiP Computer", VLSI 81, Academic Press, pp 237-246, 1981.

[U] E. Upfal, "Efficient Schemes for Parallel Communication", *Journal of ACM.*, 31(3), July 1984, pp. 507-517.

[VB] L. G. Valiant and G. J. Brebner, "Universal Schemes for Parallel Computation", *Proceedings of the 13th STOC*, 1981, pp. 263-277.

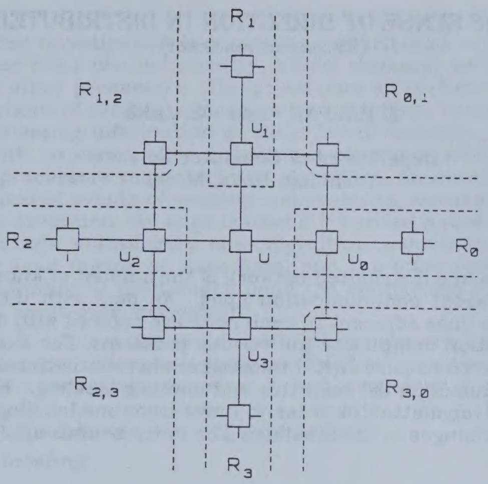
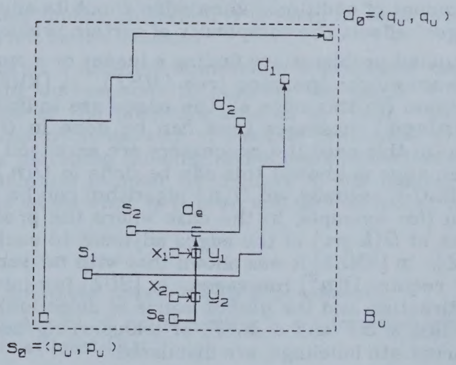

R1
R1,2
R0,1
U1
R2
U2
U
U0
R0
U3
R2,3
R3,0
R3

Figure 3.1.


d0=(qu, qu)
d1
d2
s2
de
s1
x1
y1
x2
y2
se
Bu
s0 =(pu, pu)

variable messages: (s_i, d_i), $i=0, 1, 2$

enforcer message: (s_e, d_e)

control messages: (x_1, y_1), (x_2, y_2)

Figure 4.1.

ON THE COST OF SENSE OF DIRECTION IN DISTRIBUTED NETWORKS
(Extended abstract)

Z. HARPAZ and S. ZAKS

Department of Computer Science
Technion, Haifa, Israel

ABSTRACT

Sense of direction in a distributed network is the amount of knowledge a processor has regarding its adjacent communication lines. We deal with the case where, in a complete network, the lines adjacent to each node are colored with distinct colors, and study the communication complexity for various problems. For example, a minimum spanning tree is shown to require $\Omega(n^2)$ messages, where n is the size of the network. Also studied is the connection between this and another labeling. We show that $\Omega(n^2)$ messages have to be transmitted in order to move from one labeling to another with a minimum number of changes in the labelings. The lower bounds apply also for synchronous networks.

1. INTRODUCTION

Many recent studies deal with distributed computations, where a network of processors has to solve a given problem by exchanging messages, and the total number of messages is usually regarded as the optimality criterion. Besides the problem itself, what makes these computations difficult is the amount of knowledge each processor has regarding the communication lines adjacent to itself. This notion is termed *sense of direction* in [S]. Even when the underlying graph is complete (an assumption made in many studies; see, for example, [LSP]), namely each processor is connected to all others, still the amount of additional knowledge about its adjacent edges (hidden in the labeling of the edges) affects the complexity of certain problems.

Two widely studied problems are finding a leader or a spanning tree (ST) and constructing a minimum-weight spanning tree (MST). In [KMZ1] it was shown that with *no* sense of direction (in this case all the edges are indistinguishable) finding a ST might require $\Omega(nlogn)$ messages (and can be done in $O(nlogn)$). With a *global* sense of direction (in this case the processors are arranged in a ring, and the correct orientation of each edge is known) this can be done in $O(n)$ messages, as is shown in [LMW] (see also [SSU]); actually, an $O(n)$ algorithm can be achieved with only partial sense of direction (for example, in the case where the processors are arranged in a ring and only a set of $O(logn)$ of the edges adjacent to each processor are oriented), as studied in [ASZ]. In [KMZ2] it was shown that with *no* sense of direction constructing a MST might require $\Omega(n^2)$ messages. In [SUZ] few intermediate cases (between the *no* sense of direction and the *global* sense of direction) are represented, and the complexity of finding a ST and a MST, of transferring between the models, and of achieving the appropriate labelings, are discussed.

We introduce another labeling and study the communication complexity of various problems. These include lower bounds of $\Omega(n^2)$ messages for finding a MST and for transferring between this new labeling and another one (from [SUZ]). These transformations are the main result, and the proofs involve a partition of the complete graph into complete subgraphs of size four, using a result from [HCW], while the technique used is an extension of the idea presented in [KMZ2], where a simpler partition of the complete graph is used.

2. PRELIMINARIES

The model under investigation is a complete distributed network with n processors. Each processor has a distinct identity, a local memory, and communication lines connecting it to all other processors. The processors all perform the same algorithm, that includes operations of sending a message to a neighbor, receiving a message from a neighbor and processing information in their (local) memory. We assume that the messages arrive, with no error, after a finite but otherwise arbitrary delay, and that any subset of the processors may start the algorithm. An execution of an algorithm consists of a sequence of events of sending and receiving messages. An algorithm may have more then one execution. An edge is *used* if it carried a message, and it is *unused* otherwise. The *message complexity* of an algorithm is the total number of messages the algorithm may send during an execution, and the *edge complexity* of an algorithm is the total number of used edges upon the completion of an execution of the algorithm (both are worst-case measures).

The numbers $1,2,\ldots,n$ are associated with the processors; these numbers are not known to the processors, and we use them in order to ease the discussion. With the communication line connecting processor i and processor j we associate a label $l(i,j)$, known only to processor i. Denote $N_i = \{1,2,\ldots,n\} - \{i\}$ and $N' = \{1,2,\ldots,n-1\}$. The following labelings are defined:

(1) A *complement labeling* :
for every i

$$\{\, l(i,j) \mid j \in N_i \,\} = N'$$

and for every i and j

$$l(i,j) + l(j,i) = n.$$

(2) A *matching labeling* :
for every i

$$\{\, l(i,j) \mid j \in N_i \,\} = N'$$

and for every i and j

$$l(i,j) = l(j,i)$$

(Note that this definition applies only when n is even. An extension to the odd case is straightforward and left to the reader.)

Each such labeling represents a different knowledge a processor has about the network. The matching labeling corresponds to coloring the edges of the network. The complement labeling is one of the labelings studied in [SUZ]. We deal with lower bounds for the communication complexity of various problems. In the case of the transformations between a matching labeling and a complement labeling we look for transformations with a minimum number of changes in the labels. This amounts to changing exactly half of the labels when n is even and almost half of the labels when n is odd.

3. RESULTS

We show the following for a complete network with n processors:

Theorem 1: The edge complexity of constructing a minimum-weight spanning tree, given a matching labeling, is at least $\Omega(n^2)$.

Theorem 2: The edge complexity of transforming a matching labeling into a complement labeling is at least $\Omega(n^2)$.

For the proof, we show two different matching labelings such that the sets of labels that have to be changed in order to obtain corresponding complement labelings can not be equal. Thus every transformation algorithm must distinguish between the two labelings, and in order to do so we show that it must send messages on at least $\Omega(n^2)$ edges. In a similar way we show for the reverse transformation:

Theorem 3: The edge complexity of transforming a complement labeling into a matching labeling is at least $\Omega(n^2)$.

The proofs for Theorems 2 and 3 involve a partition of the complete graph into complete subgraphs of size four, using a result from [HCW], while the technique used is an extension of the idea presented in [KMZ2].

We then show that

Theorem 4: The edge complexity of determinating whether a given labeling is a matching labeling is at least $\Omega(n^2)$.

Theorem 5: The edge complexity of constructing a matching labeling is at least $\Omega(n^2)$.

Note that, like the case in [KMZ1] and [KMZ2], all these lower bounds for the number of edges imply lower bounds for the number of messages, even when arbitrary length messages and a synchronous execution are allowed. Note also that $O(n^2)$ algorithms for all the above problems are simple, thus proving the optimality of these bounds.

REFERENCES

[ASZ] H. Attiya, N. Santoro and S. Zaks, *From rings to complete networks: a case study in distributed algorithms*, in preparation.

[HCW] H. Hanani, D.K.R. Chaundhuri and R.M. Wilson, *On resolvable designs*, **Discrete Mathematics** 3, 1972, pp. 343-357.

[KMZ1] E. Korach, S. Moran and S. Zaks, *Tight lower and upper bounds for distributed algorithms for a complete network of processors*, **Proceedings of the 3rd ACM Symposium on Principles of Distributed Computing**, Vancouver, Canada, August 1984, pp. 199-207.

[KMZ2] E. Korach, S. Moran and S. Zaks, *The optimality of distributive constructions of minimum weight and degree restricted spanning trees in a complete network of processors*, **Proceedings of the 4th ACM Symposium on Principles of Distributed Computing**, Minaki, Canada, August 1985, pp. 277-286.

[LMW] M. C. Loui, T. A. Matsushita and D. B. West, *Election in a complete network with a sense of direction*, to appear in the **Information Processing Letters**.

[LSP] L. Lamport, R. Shostak and M. Pease, *The byzantine generals problem*, **ACM Transactions on Programming Languages and Systems**, 4, 3, 1982, pp. 382-401.

[S] N. Santoro, *Sense of direction, topological awareness and communication complexity*, Report SCS-TR-52, School of Computer Science, Carleton University, Ottawa, Canada, May 1984.

[SSU] J. Sack, N. Santoro and J. Urrutia, *O(n) election algorithms in complete networks with global sense of orientation*, Report SCS-TR-49, School of Computer Science, Carleton University, Ottawa, Canada, May 1984.

[SUZ] N. Santoro, J. Urrutia and S. Zaks, *Sense of direction and communication complexity in distributed networks*, to appear in the **Proceedings of the International Workshop on Distributed Algorithms on Graphs**, Carleton University, Ottawa, Canada, August 1985.

AN OPTIMAL SHOUT-ECHO ALGORITHM FOR SELECTION IN DISTRIBUTED SETS †

JOHN M. MARBERG and ELI GAFNI
Computer Science Department
University of California
Los Angeles, CA 90024

ABSTRACT

In a Shout-Echo network, processors communicate by means of broadcast messages (shout) and replies to received messages (echo). Performance is measured in terms of communication cycles, where a cycle consists of one broadcast message and the replies to it from all other processors. In this paper we consider the problem of selection by rank in a set distributed among the processors of a Shout-Echo network. We show a tight lower and upper bound of size $\Theta(\log \min \{n_{\max 2}, k\})$ on the number of cycles required for selection, where $n_{\max 2}$ is the size of the second-largest subset of elements in any processor, and k is the rank of the element to be selected. We first show the lower bound, then give an efficient algorithm which achieves this bound in a wide range of cases. The algorithm improves the previous best upper bound by a factor of $\log p$, where p is the number of processors in the network.

1. INTRODUCTION

A *Shout-Echo* network [Sant83b] is a distributed computation model in which a collection of independent processors communicate by means of broadcast messages (shout) and replies to received broadcast messages (echo). The performance of algorithms in this model is measured in terms of the total number of *communication cycles* required by the computation, where a communication cycle consists of one broadcast message and the replies to it from all other processors. Each message is assumed to contain at most a constant number of elementary units of information.

In this paper we consider the problem of selecting an element by rank from a totally ordered set distributed among the processors of a Shout-Echo network. The selection problem has received considerable attention, partly because it is a subproblem of sorting, which is among the most fundamental problems in computing. The applications of selection are mainly in the area of order statistics. For example, quantile analysis, and finding the median. Another type of application arises in operations research, namely the computation of optimum distribution of effort [Gali79]. Sequential (single-processor) selection algorithms with linear running time are given in [Blum73, Scho76]. Efficient selection algorithms have also been designed for various decentralized computing environments, such as the PRAM shared memory model [Vish83], and communication networks with various topologies [Fred83, Rode84].

---

† This research was supported by an IBM Faculty Development Award.

In the context of the Shout-Echo model, the selection problem was previously investigated by Santoro and Sidney [Sant82, Sant83b, Sant83c], who presented two different selection algorithms which run in $O(\min\{k, p\log\frac{k}{p}\})$ cycles, where k is the rank of the element to be selected and p is the number of processors in the network. They also showed the existence of a lower bound of $\Omega(\log k)$ cycles when the elements are evenly distributed (i.e., all processors have the same number of elements) and $k \le \frac{n}{p}$, where n is the total number of elements in the network. The lower bound is a generalization of the $\Omega(\log n)$ bound on finding the median when $p=2$, shown by Rodeh [Rode82]. Rotem, Santoro and Sidney [Rote83] gave an $O(\log p \log\frac{n}{p})$ algorithm for finding the median of an evenly distributed set. This algorithm can be easily generalized for arbitrary (uneven) distributions and for any rank k, thereby improving the upper bound of the general problem to $O(\log p \log \min\{n_{\max 2}, k\})$, where $n_{\max 2}$ is the size of the second-largest subset of elements in any processor.

Our work improves on the above results in the following ways. First, we generalize the lower bound of [Sant83b], proving that $\Omega(\log \min\{n_{\max 2}, k\})$ cycles are required for any rank k and any distribution of the elements. We then present a selection algorithm which runs in $O(\log k)$ cycles for $k \le n_{\max 2}$, and otherwise in $O(\log n)$ cycles. By showing that in a wide range of cases $O(\log n) = O(\log n_{\max 2})$ we prove that the upper and lower bounds on selection in the Shout-Echo model are tight. Our algorithm thus improves the previous best upper bound by a factor of $\log p$.

The remainder of this paper is organized as follows. Section 2 defines our notation. The lower bound is shown in Section 3. The algorithm is presented in Section 4, and its complexity analysis in Section 5. Concluding remarks are given in Section 6.

2. DEFINITIONS AND NOTATION

The p processors of a Shout-Echo network are denoted $P_1, P_2, \ldots, P_p$, where each P_i is a unique identifier taken from a totally ordered set of identifiers.

Let N be a collection of n elements distributed arbitrarily among the processors. Without loss of generality (w.l.g.) we may assume that N is a set, i.e., that all elements in N are distinct. If not, we can replace each element ξ at P_i with the triple (ξ, P_i, j_ξ) where j_ξ is a unique index within P_i, and use lexicographic order among the triples. We denote the j'th largest element in N as $N[j]$. The subset of elements at processor P_i is denoted N_i, with cardinality $|N_i| = n_i$. We assume $n_i > 0$. The terms $n_{\max}$ and $n_{\max 2}$ denote, respectively, the largest and the second-largest n_i. $N_i[j]$ denotes the j'th largest element in N_i.

Selection is the task of identifying $N[k]$ for a given rank k. W.l.g. we may assume $1 \le k \le \lceil\frac{n}{2}\rceil$ (if not, reverse the total order and select the element of rank $n-k+1$).

Throughout the paper we use log to denote logarithm of base 2.

3. LOWER BOUNDS

To show the lower bounds we use an adversary argument adapted from Frederickson [Fred83]. The argument applies only to comparison-based algorithms, i.e., algorithms in which the values of the elements are used only in comparisons with each other.

Theorem 1. Let $n_{\max 2} \le k \le \lceil \frac{n}{2} \rceil$. The number of communication cycles required to select the k'th largest element in a Shout-Echo network is $\Omega(\log n_{\max 2})$.

Proof. We devise an adversary that, given the cardinalities n_i and a selection algorithm, provides input sets N_i such that the algorithm requires $\Omega(\log n_{\max 2})$ cycles when executed on that input.

The adversary is free to make each element arbitrary large or small, as long as the relative order in each N_i is maintained consistently. Initially, none of the elements has a fixed value. The adversary follows the execution of the algorithm, fixing the value of elements as the algorithm proceeds. Elements not yet fixed are candidates for selection. Fixed elements are made either "very large" or "very small", in the sense that they are larger or smaller than all the remaining candidates in the network. By keeping fewer than k very large elements and fewer than $n-k$ very small elements at all times, the adversary excludes such elements from being selected. Clearly, the algorithm cannot terminate before the number of candidates is reduced to one. Total order is maintained among the fixed elements of all processors by making each new very large element (very small element, respectively) smaller (larger) than all the existing very large (very small) elements.

Let $P_{\max}$ and $P_{\max 2}$ denote two processors which have, respectively, $n_{\max}$ and $n_{\max 2}$ elements. The adversary initializes the elements of N as follows. In each of $P_{\max}$ and $P_{\max 2}$, $n_{\max 2}$ elements are made candidates. Among the remaining $n-2n_{\max 2}$ elements, an arbitrary $k-n_{\max 2}$ elements are made very large and the rest very small. With this setup, selecting $N[k]$ reduces to finding the median of the $2n_{\max 2}$ candidates.

Whenever a message is sent which contains a candidate of $P_{\max}$ that is larger or equal (smaller, respectively) than the median of the candidates of $P_{\max}$, the adversary fixes that candidate and all those larger (smaller) than it at $P_{\max}$ to be very large (very small), and an equal number of candidates at $P_{\max 2}$ to be very small (very large). From that point on, these elements are no longer candidates. When the message contains a candidate of $P_{\max 2}$, the same action with the roles of $P_{\max}$ and $P_{\max 2}$ reversed is taken. No action is taken by the adversary if the message does not contain a candidate. Concurrent messages are handled in some arbitrary order.

Let $2m$ be the number of remaining candidates immediately before a message containing one of the candidates is sent. It can be seen that no more than $m+1$ candidates are fixed by the adversary as a result of that message. In other words, the number of candidates in the network is reduced by a factor of at most $\frac{3}{4}$ after each message. As there are initially $2n_{\max 2}$ candidates, $\Omega(\log n_{\max 2})$ messages containing candidates are necessary in order to reduce the number of candidates to one. Since all the candidates are located at $P_{\max}$ and $P_{\max2}$, at most two such messages can be sent in each cycle, thus requiring $\Omega(\log n_{\max2})$ cycles. ■

We now show a lower bound for the case $k \le n_{\max2}$.

Theorem 2. Let $k \le n_{\max2}$. The number of communication cycles required to select the k'th largest element in a Shout-Echo network is $\Omega(\log k)$.

Proof. We use the same adversary argument as in Theorem 1. The only difference is in the initialization of candidates and fixed elements. Here, only k elements in each of $P_{\max}$ and $P_{\max2}$ are made candidates, and all the remaining elements in the network are made very small. The result follows from an analysis similar to that of Theorem 1. ■

Corollary 1. The number of communication cycles required to select the k'th largest element in a Shout-Echo network is $\Omega(\log \min \{k, n_{\max2}\})$. ■

4. THE SELECTION ALGORITHM

A naive approach to selection is to sort all the elements, then retrieve the selected element directly by rank. This, however, is inefficient because the extra information provided by sorting comes at a cost and is not really needed. A more promising approach is the following. Repetitively reduce the number of candidates for selection by applying some filtering mechanism. When the number of remaining candidates gets below a specified threshold value, sort the remaining candidates and retrieve the selected element by rank. In the sequel we present a selection algorithm which follows this approach. The algorithm works for an arbitrary distribution of the elements among the processors. We first give a general description of the algorithm, then discuss the implementation in the Shout-Echo model. A complexity analysis is given in Section 5.

4.1. Description of the Algorithm

The algorithm consists of three stages: initialization (step 1 in the description below); iterative filtering (steps 2-6); and termination (step 7). In each filtering iteration we locate among the remaining candidates an element which we call a *filter*, and find out its rank. Depending on the relative order between this rank and k, we then purge either all the candidates that are smaller or equal to the filter, or those that are larger or equal. The filters we use are such that a large portion of the remaining candidates is purged in each iteration, thus the threshold value of the number of candidates is reached quickly. In the termination stage, all the remaining elements are collected into one processor, and the selection is completed locally.

We denote the number of remaining candidates for selection at each point in the algorithm by m. The subset of candidates at each P_i is denoted M_i, with cardinality $|M_i| = m_i$. The threshold value is denoted m^*.

Algorithm Select(k)

```
(1) $M_i := N_i$ ; $m_i := n_i$ ; $m := \sum_{i=1}^{p} m_i$ .
(2) Find the median $M_i[\lceil \frac{m_i}{2} \rceil]$ of each $M_i$ .[†] Denote this element $med_i$ .
    If $M_i$ is empty, $med_i$ is given a dummy value.
(3) Sort the pairs $(med_i , m_i)$ in descending order of the first coordinate.
    Let this order be $(med_{i_1}, m_{i_1}) \geq (med_{i_2}, m_{i_2}) \geq \cdots \geq (med_{i_p}, m_{i_p})$ .
(4) Find the smallest index $l$ such that $\sum_{j=1}^{l} m_{i_j} \geq \frac{m}{2}$. Denote $med_{i_l}$ as $med_{1/2}$.[‡]
(5) Find the total number of candidates that are greater or equal to $med_{1/2}$ .
    Denote this number $m_{1/2}$ .
(6) Compare $m_{1/2}$ with $k$ and choose the appropriate case below.
    Case 1. $m_{1/2} = k$
        The selected element is $med_{1/2}$ ; terminate the algorithm.
    Case 2. $m_{1/2} > k$
        (a) Purge from each $M_i$ the elements smaller or equal to $med_{1/2}$ .
        (b) $m_i := | M_i |$ ; $m := m_{1/2} - 1$ .
        (c) If $m > m^*$ go to step 2; otherwise go to step 7.
    Case 3. $m_{1/2} < k$
        (a) Purge from each $M_i$ the elements greater or equal to $med_{1/2}$ .
        (b) $m_i := | M_i |$ ; $m := m - m_{1/2}$ ; $k := k - m_{1/2}$ .
        (c) If $m > m^*$ go to step 2; otherwise go to step 7.
(7) (a) Collect all the remaining candidates into one local set $M$ .
    (b) Find $M[k]$, which is the selected element; terminate the algorithm.
```

We now show that the algorithm works correctly. Assume inductively that at the beginning of the current iteration of the filtering stage (steps 2-6) the element to be selected has not been purged, and that k is the correct rank of this element among the remaining candidates. This is clearly true at the beginning of the algorithm.

[†] This can be done using some efficient sequential selection algorithm (e.g., [Blum73]) or by sorting.

[‡] $med_{1/2}$ is the filter element. Its choice will be explained in Section 5.

In case 1 of step 6, the number of candidates greater or equal to $med_{1/2}$ is found to be k, thus the decision to select $med_{1/2}$ is correct. In case 2, since $m_{1/2} > k$, the element we wish to select is strictly greater than $med_{1/2}$ (remember that w.l.g. all elements are distinct). Thus, all candidates smaller or equal to $med_{1/2}$ can be purged. In case 3 a similar argument applies, except that since the $m_{1/2}$ candidates being purged are greater than the selected element, the rank k has to be reduced by the same quantity. Since at least one candidate is purged each time case 2 or 3 occurs, m becomes smaller and smaller, and eventually the algorithm will either terminate in case 1 of step 6, or reach step 7 where the correct element is selected by collecting all the remaining candidates into a local set M and finding $M[k]$.

4.2. Implementation in the Shout-Echo Model

Implementing the above algorithm in the Shout-Echo model is straightforward. An arbitrary processor is designated as "supervisor". A computation step that requires local data of more than one processor is implemented as follows. The supervisor sends a broadcast message requesting the data, and each processor replies by sending its local data. The supervisor then performs the computation locally and broadcasts the result.

We now give the details of each step. W.l.g. assume that the supervisor is P_1. In step 1, P_1 collects m_i from all the processors, then computes m and broadcasts it. Step 2 is performed locally at each processor. In step 3, P_1 collects all the pairs (med_i, m_i) and sorts them. In step 4, P_1 finds $med_{1/2}$ and broadcasts it. In step 5, each P_i finds the number of candidates in M_i that are greater or equal to $med_{1/2}$. These numbers are then sent to P_1, who adds them up and broadcasts the sum $m_{1/2}$. Step 6 is performed locally at each processor. In step 7, all the remaining candidates are sent to P_1. This may require several cycles, where in each cycle each processor sends one element. When all candidates have been collected, P_1 completes the selection locally and broadcasts the result.

5. COMPLEXITY ANALYSIS

To find the cycle complexity of the algorithm, we need to determine the number of iterations performed in the filtering stage, and calculate the cost of each step of the computation.

The analysis of the number of filtering iterations is illustrated by Figure 1, which captures the situation at the beginning of a typical iteration. The sets of candidates are shown in the order $M_{i_1}, M_{i_2}, \ldots, M_{i_p}$ from left to right, where M_{i_j} corresponds to the pair (med_{i_j}, m_{i_j}). The elements in each set are shown in descending order from top to bottom. Since the sets are ordered in descending order of the medians, it can be seen that for any set M_{i_j}, half the candidates in M_{i_j} and at least half the candidates in each set to the right of M_{i_j} are smaller or equal to med_{i_j}. Similarly, half the set M_{i_j} and at least half of each set to the left of M_{i_j} are greater or equal to med_{i_j}. This is shown in Figure 1 by the encircled areas.

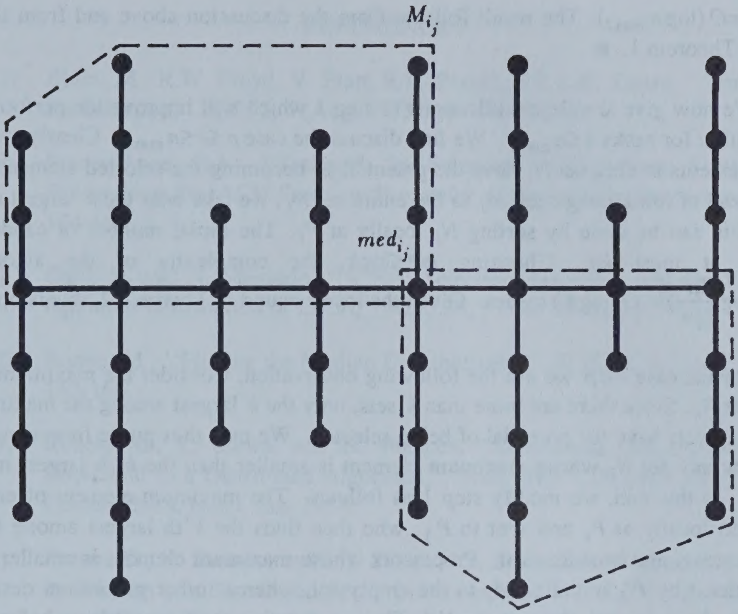


Figure 1: The Filtering Stage

In particular, since $med_{1/2}$ $(=med_{i_l})$ was chosen such that $\sum_{j=1}^{l} m_{i_j}$ is the smallest partial sum of cardinalities which is greater or equal to $\frac{m}{2}$, it can be seen that at least $\frac{m}{4}$ candidates are smaller or equal to $med_{1/2}$, and at least $\frac{m}{4}$ candidates are greater or equal to $med_{1/2}$. Consequently, in each occurrence of case 2 or 3 in step 6 at least one fourth of the remaining candidates are purged. Thus, $O(\log \frac{n}{m^*})$ iterations suffice in order to reduce the number of candidates below m^*.

It can be easily seen that each of steps 1-6 requires a constant number of cycles. The number of cycles in step 7 is bounded by $O(m^*)$. The total complexity of the selection algorithm is therefore $O(m^* + \log \frac{n}{m^*})$ cycles. Choosing $m^* \leq \log n$, the complexity is $O(\log n)$ cycles. The following corollary shows that this is optimal in a wide range of cases.

Corollary 2. Let $k \geq n_{\max 2}$. Given a constant $0 < \varepsilon \leq 1$ such that $n_{\max 2} \geq \varepsilon p$ and $n_{\max} \leq \frac{n_{\max 2}^2}{\varepsilon}$, the complexity of selecting the k'th largest element in a Shout-Echo network is $\Theta(\log n_{\max 2})$ cycles.

Proof. $n_{\max 2} \geq \varepsilon p$ and $n_{\max} \leq \frac{n_{\max 2}^2}{\varepsilon}$ imply $n \leq (p-1) n_{\max 2} + n_{\max} \leq \frac{2 n_{\max 2}^2}{\varepsilon}$, and hence

$O(\log n)=O(\log n_{\max 2})$. The result follows from the discussion above and from the lower bound of Theorem 1. ■

We now give simple modifications to step 1 which will improve the performance of the algorithm for ranks $k \le n_{\max 2}$. We first discuss the case $p \le k \le n_{\max 2}$. Clearly, only the k largest elements in each set N_i have the potential of becoming the selected element. Therefore, instead of initializing each M_i to the entire set N_i, we take only the k largest elements of N_i. This can be done by sorting N_i locally at P_i. The initial number of candidates is therefore at most kp. Choosing $m^* \le \log k$, the complexity of the algorithm is $O(m^* + \log \frac{kp}{m^*}) = O(\log k)$ cycles. Given the lower bound of Theorem 2, this is optimal.

For the case $k<p$ we use the following observation. Consider the maximum element in each set N_i. Since there are more than k sets, only the k largest among the maximum elements of all sets have the potential of being selected. We may thus purge from candidacy in its entirety any set N_i whose maximum element is smaller than the k'th largest maximum element. To this end, we modify step 1 as follows. The maximum element of each N_i is determined locally at P_i and sent to P_1, who then finds the k'th largest among the maximum elements and broadcasts it. Processors whose maximum element is smaller than the one broadcast by P_1 initialize M_i to the empty set, whereas other processors designate as candidates the k largest elements in N_i. The computation then proceeds as before. Since now only k of the processors have candidates, the initial number of candidates is at most k^2. Choosing $m^* \le \log k$, the complexity of the algorithm is $O(m^* + \log \frac{k^2}{m^*}) = O(\log k)$ cycles, which by Theorem 2 is optimal.

Corollary 3. Let $k \le n_{\max 2}$. The complexity of selecting the k'th largest element in a Shout-Echo network is $\Theta(\log k)$ cycles. ■

6. CONCLUSIONS

We have discussed the complexity of selection by rank in Shout-Echo networks. We have first shown a lower bound, then given an algorithm which achieves this bound in a wide range of cases, thus proving that the bound is tight. The algorithm improves by a factor of $\log p$ the previous best upper bound, derived from the work in [Rote83].

In [Sant83a], Santoro, Scheutzow and Sidney presented a probabilistic shout-echo selection algorithm in which the filter element is chosen at random, and showed that the expected complexity of the algorithm is $O(\log n)$ cycles. The worst-case behavior of our algorithm compares favorably with the average-case behavior of this algorithm.

Santoro and Sidney [Sant83c] showed that for a threshold of $m^*=O(p)$, the expected number of filtering iterations in their selection algorithm is $O(\log \log k)$. In [Sant83a] it is shown that by combining the algorithm of [Sant83c] with the probabilistic algorithm of [Sant83a] an expected complexity of $O(\log p + \log \log k)$ cycles is attained. It is interesting to see how an average-case analysis of our algorithm would compare with this result.

REFERENCES

[Blum73] Blum, M., R.W. Floyd, V. Pratt, R.L. Rivest, and R.E. Tarjan, ''Time Bounds for Selection,'' *JCSS* 7, 4 (Aug. 1973), pp. 448-461.

[Fred83] Frederickson, G.N., ''Tradeoffs for Selection in Distributed Systems,'' in *Proceedings 2nd ACM Symp. on Principles of Distributed Computing*, 1983, pp. 154-160.

[Gali79] Galil, Z. and N. Megiddo, ''A Fast Selection Algorithm and the Problem of Optimum Distribution of Effort,'' *JCSS* **26**, 1 (Jan. 1979), pp. 58-64.

[Rode82] Rodeh, M., ''Finding the Median Distributively,'' *JCSS* **24**, 2 (April 1982), pp. 162-166.

[Rode84] Rodeh, M., L. Shrira, and N. Francez, ''Distributed K-Selection: From a Sequential to a Distributed Algorithm,'' Tech. Rep. 128, IBM Israel Scientific Center, Haifa, Israel, Jan. 1984.

[Rote83] Rotem, D., N. Santoro, and J.B. Sidney, ''A Shout-Echo Algorithm for Finding the Median of a Distributed Set,'' in *Proceedings 14th S.E. Conf. on Combinatorics, Graph Theory and Computing*, Boca Raton, FL, 1983, pp. 311-318.

[Sant82] Santoro, N. and J. Sidney, ''Order Statistics on Distributed Sets,'' in *Proceedings 20th Allerton Conf. on Communication, Control and Computing*, Univ. of Illinois at Urbana Champaign, 1982, pp. 251-256.

[Sant83a] Santoro, N., M. Scheutzow, and J. B. Sidney, ''A Reduction Technique for Distributed Selection: II,'' Tech. Rep. SCS-TR-35, School of Computer Science, Carleton Univ., Ottawa, Canada, Dec. 1983.

[Sant83b] Santoro, N. and J.B. Sidney, ''Communication Bounds for Selection in Distributed Sets,'' Working Paper 83-39, Faculty of Administration, Univ. of Ottawa, Ottawa, Canada, 1983.

[Sant83c] Santoro, N. and J. B. Sidney, ''A Reduction Technique for Distributed Selection: I,'' Tech. Rep. SCS-TR-23, School of Computer Science, Carleton Univ., Ottawa, Canada, April 1983.

[Scho76] Schonhage, A., M. Paterson, and N. Pippenger, ''Finding the Median,'' *JCSS* **13**, 2 (Oct. 1976), pp. 184-199.

[Vish83] Vishkin, U., ''An Optimal Parallel Algorithm for Selection,'' Ultra Computer Note 64, Tech. Rep. 106, Dept. of Computer Science, Courant Inst. of Math. Sciences, New York Univ., New York, NY, Dec. 1983.

AN APPROACH TO LANDMARK-BASED SHAPE RECOGNITION [†]

NIRWAN ANSARI and EDWARD J. DELP
Purdue University
School of Electrical Engineering
West Lafayette, Indiana 47907
(317)-494-1740

ABSTRACT

In this paper, we introduce a new method of comparing two-dimensional shapes based on landmarks or tokens derived from the shape contour. This approach is motivated by the difficulties associated with shape recognition of an object when the segmentation process fails to produce the entire object contour (or silhouette) due to errors in the segmentation algorithm or object occlusion. We derive a measure of similarity for objects with three landmarks. For objects with more than three landmarks, we develop a method based on two statistical shape measures and another method using a MMSE (minimum mean square error) affine transformation to differentiate shapes.

I. INTRODUCTION

Classification of objects based on shape information is a fundamental problem in computer vision. Segmentation which results in the entire closed contour or silhouette of an object is a necessary prerequisite for many shape analysis methods [8] such as moment descriptors, Fourier descriptors, and template matching [5][11][10]. Complete segmentation of an object is usually difficult, if not impossible, in natural scenes due to noise or occluded objects. In this paper we present a new approach to shape recognition based on landmarks or tokens of an object. Landmarks are points of interest of the object which have important shape attributes. They are usually the extremals such as corners, holes, and points with high curvature, they can also be problem specific based on a-priori knowledge. This concept was proposed in the field of biology, known as morphometrics [1] [2] [3], to measure growth and shape change. D'Arcy Thompson [9] pioneered the concept of a cartesian co-ordinate transformation to describe shape deformation in the early 20th century. He stated that shape change should be studied by means of a transformation of one object to another. Since we are concerned with landmarks, we shall concentrate on the study of transformations of geometric shapes formed from the landmarks. In Section II, we discuss the representation of shapes defined by three landmarks, i.e. triangles, and the shape preserving properties of affine transformations. In Section III, we describe method for shape classification based on the landmarks. For objects with three landmarks, we will use a measure of similarity to classify objects. We then develop two methods to classify objects with more than three landmarks. In the first method, we derive two statistical shape measures. Finally, we describe a method involving a MMSE (minimum mean square error) affine transformation. In this paper we shall not discuss how landmarks are chosen, however we will demonstrate the relative robustness of our shape measures.

II. REPRESENTATION OF TRIANGULAR DEFORMATION

Growth of a biological object can be observed by noting certain critical points (landmarks) of the object after a period of time. For an object of three landmarks, the growth change is similar to the deformation of the triangle formed from the landmarks. Bookstein proposed the representation of a deformation from one trian-

---

[†] This research was partially supported by the National Institutes of Health under Grant No. 1RO1 HL29716-01A1.

gle to another by a homogeneous transformation that maps the inscribed circle of one triangle to the inscribed ellipse of the other triangle [2][3]. From the ellipse and the circle, he defined two measures, the principals of dilatation, which are the ratio of the major and minor axes of the transformed ellipse to the diameter of the circle respectively. We shall call these quantities the major and minor dilatation (also known as the strain tensors [4]), denoted by D_L and D_S respectively (see Figure 1). He further defined a measure of directionality (we shall define it as a measure of shape similarity) known as anisotropy, denoted by a, as the quotient of the major and the minor dilatation, and a measure of size change, denoted by s, as their product. We summarize the above by letting (see Figure 1):

r = length of the radius of the inscribed circle
α = length of the minor axis of the inscribed ellipse
β = length of the major axis of the inscribed ellipse

then,

$$D_L = \frac{\alpha}{2r} \quad \text{is the major dilatation} \tag{1}$$

$$D_S = \frac{\beta}{2r} \quad \text{is the minor dilatation} \tag{2}$$

$$a = \frac{D_L}{D_S} \quad \text{is the anisotropy} \tag{3}$$

$$s = D_L D_S \quad \text{is the size change} \tag{4}$$

We can then implement the above notion of shape change by means of an affine transformation. Let the point (u,v) be the image of the point (x,y) under an affine transformation. Therefore, we know that:

$$u = ax + by + c \tag{5a}$$

$$v = dx + ey + f \tag{5b}$$

where a, b, c, d, e, and $f \in \mathbf{R}$

$$\delta = ae - bd \neq 0$$

It can be shown that an affine transformation [6][7] is an one-to-one mapping of the plane onto itself with the following properties:

- Collinearity is preserved under the transformation.
- Noncollinearity is preserved under the transformation.
- Betweenness relation is preserved under the transformation, i.e. a mid-point of a line remains the mid-point of the transformed line.

Thus, a point is mapped into a point, a line into a line, and an angle remains an angle. Since there are six coefficients which describe an affine transformation, they are uniquely determined if three noncollinear points and their corresponding noncollinear image points are known. By direct substitution of a circle equation into an affine transformation, it is readily seen that the transformed circle is an ellipse. It can also be shown that the area of a triangle under an affine transformation is multiplied by $|\delta|$ [6]. An affine transformation satisfies the above notion of a homogeneous transformation that maps a circle into an ellipse. Figure 1 shows different representations of a deformation from one triangle to another. Coefficients of the unique affine transformation which maps one triangle into another are computed using the following equation, (see Figure 1):

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = A^{-1} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} d \\ e \\ f \end{bmatrix} = A^{-1} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{6}$$

$$\text{where} \quad A = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix}$$

where points (u_i, v_i) are the image of the points (x_i, y_i), i=1,2,3 under the transformation described by Equation 5. Since vertices of a triangle are noncollinear, $\det(A) \neq 0$ and A^{-1} exists. Assuming we know the vertices of the first triangle, we can compute the parameters (radius and center) of the inscribed circle. The inscribed circle is then mapped through the transformation into an ellipse, and the measures described in Equations 1, 2, 3 and 4 can be computed.

III. OBJECTS CLASSIFICATION

In this section we describe how the affine transformation described above can be used for shape recognition. To identify an observed (unknown) object from a library of objects, we will find the necessary transformation that maps landmarks of each object in the library to corresponding landmarks of an observed (unknown) object. This approach is different than that proposed by Bookstein because we are interested in shape recognition and not shape change. From the landmarks of an object, we form a polygon (i.e. a triangle for an object with three landmarks) by joining the landmarks with straight lines. Thus, an object is characterized by the polygon formed from its landmarks. For objects with three landmarks, we evaluate the anisotropy. For objects with more than three landmarks we will need to extend our concept of similarity.

A. Objects with Three Landmarks

Each object is characterized by a triangle formed by joining its three landmarks with straight lines. From Section II, we know that there exists an affine transformation that maps one triangle to another. We find the affine transformation that maps the triangle formed from the three landmarks of each library object to that of the observed object. We then compute the anisotropy of each transformation. If a library object is similar to the observed object, the inscribed circle of the library object (triangle) is transformed into an inscribed circle of the observed object (triangle), and thus yields an anisotropy of value 1. The anisotropy therefore is a measure of how similar the two objects are. Classification is achieved by choosing the library object which has a value of anisotropy closest to 1. Figure 2 demonstrates this approach for a library of three objects. The observed object, which is object 1, is correctly identified. Let:

N be the number of landmarks of an observed object.

p be the perimeter of the polygon (a triangle for three landmarks) formed from landmarks of the observed object.

therefore

$\lambda = \frac{p}{N}$ is the average side length of the polygon,

and let

$\mathbf{n}_\gamma$ be a zero-mean uniformly distributed random variable with density $p_{\mathbf{n}_\gamma}(\mathbf{n}_\gamma) = \frac{10}{\lambda\gamma} rect(\frac{\mathbf{n}_\gamma}{0.1\lambda\gamma})$.

The parameter γ is a non-zero integer that controls the standard deviation of $\mathbf{n}_\gamma$ which can be related to a percentage of the average side length λ. Figures 3, 4, and 5 demonstrate the relative robustness of this technique. In these figures, we introduce uncertainty in the location of some of the co-ordinates of the observed object, (u_i, v_i), i=1,2,3, by adding $\mathbf{n}_\gamma$ to the co-ordinates. This simulates the situation where the location of the landmarks is incorrectly chosen by the landmark detector. Notice that the observed object can be correctly determined by the anisotropy measure. Figure 5 demonstrates the classification performance of this shape analysis technique when all three landmarks are noisy.

Since the betweenness relation is preserved under the affine transformation, scale change and anisotropy are invariant when the observed object is rotated or translated.

B. Objects with more than three landmarks

We again form a polygon by connecting the landmarks of an object with straight lines. We develop two methods to classify objects with more than three landmarks. In the first method, we define two statistical shape measures based on the scale change property. We also define another similarity measure we call the MMSE (minimum mean square error) affine transformation.

Method 1.

We partition each object (polygon) into triangles. We find the affine transformation which maps each partitioned triangle of the objects in the library to the corresponding triangle of the observed (unknown) object. Let:

M be the total number of partitioned triangles of each object in the library as well as the observed object.

$\underset{\sim}{\Delta_i}$ be the ith triangle of an library object (polygon).

Δ_i be the corresponding ith triangle of the observed object.

D_{L_i} be the major dilatation under the transformation from Δ_i to $\tilde{\Delta}_i$.

D_{S_i} be the minor dilatation under the transformation from Δ_i to $\tilde{\Delta}_i$.

and let:

$$a_i = \frac{D_{L_i}}{D_{S_i}}, \tag{7}$$

$$s_i = D_{L_i} D_{S_i}, \tag{8}$$

where a_i and s_i are the anisotropy and the scale change obtained from the transformation of Δ_i to Δ_i, respectively.
We also compute the average scale change:

$$\bar{s} = \frac{1}{M}\sum_{i=1}^{M} s_i, \tag{9}$$

If the observed object is similar to a library object, their corresponding Δ_i's and Δ_i's are similar for all i=1,...,M. Each transformation from Δ_i to Δ_i should then yield the same scale change and anisotropy of value 1. We define two statistical shape measures:

$$\sigma_s^2 = \frac{1}{M}\sum_{i=1}^{M}(s_i-\bar{s})^2 \tag{10}$$

$$\sigma_a^2 = \frac{1}{M}\sum_{i=1}^{M}(a_i-1)^2 \tag{11}$$

σ_s is the sample standard deviation of the scale change for the triangles. If the observed object and the library object are similar, the average anisotropy is 1. Therefore, σ_a is the standard deviation of anisotropy from the average anisotropy, i.e. the deviation from similarity. Classification is achieved by choosing the library object that yields the least values for both σ_s and σ_a. By the same reasons mentioned in Section III.A, the derived parameters σ_s and σ_a are also invariant to rotation and translation. Figure 6 shows that the observed object is correctly identified by this technique. We may partition a polygon in many ways, as long as the partitioning of the library polygon and the observed polygon are consistent, objects can be classified in the same manner (see Figure 7). Figures 8 and 9 demonstrate the relative robustness of this technique. In Figure 8, we introduce noise in the landmarks. The noise is defined similarly to that described in Section III.A. In Figure 9, one landmark of the observed object is missing. This corresponds to a situation where the landmark detector cannot find a landmark.

Method 2.

As discussed above, an affine transformation has shape-preserving properties, i.e. a triangle is mapped into a triangle, a polygon into a polygon. However, for objects with more than three landmarks, there is no guarantee that there exists an affine transformation that would map all the landmarks of a library object to all the landmarks of the observed object. We propose an affine transformation that estimates the landmarks of an observed object from those of a library object in the minimum mean-square sense. We first translate the centers of mass of both the observed object and the library object to the origin. Let:

N be the total number of landmarks.
(x_i, y_i) and (u_i, v_i) be the corresponding landmarks of the library and the observed object after the translation mentioned above.

We find an affine transformation

$$\hat{u} = ax + by + c \qquad (12)$$

$$\hat{v} = dx + ey + f$$

such that

$$\epsilon = \sum_{i=1}^{N} e_{u_i}^{2} + e_{v_i}^{2} \qquad (13)$$

$$\text{where } e_{u_i} = \left| \hat{u}_i - u_i \right| = \left| ax_i + by_i + c - u_i \right|$$

$$e_{v_i} = \left| \hat{v}_i - v_i \right| = \left| cx_i + ey_i + f - v_i \right|$$

is minimized. By taking the partial derivatives of Equation 13 with respect to each coefficient of the affine transformation described by Equation 12, we obtain:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = B^{-1} \begin{bmatrix} \sum_{i=1}^{N} x_i u_i \\ \sum_{i=1}^{N} y_i u_i \\ \sum_{i=1}^{N} u_i \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} d \\ e \\ f \end{bmatrix} = B^{-1} \begin{bmatrix} \sum_{i=1}^{N} x_i v_i \\ \sum_{i=1}^{N} y_i v_i \\ \sum_{i=1}^{N} v_i \end{bmatrix} \qquad (14)$$

$$\text{where } B = \begin{bmatrix} \sum_{i=1}^{N} x_i^{2} & \sum_{i=1}^{N} x_i y_i & \sum_{i=1}^{N} x_i \\ \sum_{i=1}^{N} x_i y_i & \sum_{i=1}^{N} y_i^{2} & \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} y_i & 1 \end{bmatrix}$$

Notice that elements of matrix B are in terms of the co-ordinates of the library object. An affine transformation preserves shape, and hence the "estimated polygon" which results from the above mapping of a particular library object will be a "similar type" of polygon. This property could be used for object classification.

Let:

A_o be the area of the observed polygon

A_t be the area of the polygon which results from the MMSE transformation of a particular library shape:

A reasonable measure of error would be proportional to A_ρ where

$$A_\rho = (A_o \cup A_t) \backslash (A_o \cap A_t) \tag{15}$$

A_ρ is thus a measure of deviation from similarity. Classification of an object can be achieved by choosing the library object that yields the least A_ρ. In Figure 10, we can identify the observed object as object 1 since A_ρ is zero. In Figure 11, all the landmarks of the observed object are noisy. Notice that A_ρ has the smallest value for object 1 in this case also.

IV. CONCLUSION

We have introduced and developed methods for comparing shapes based on landmarks. Our approach has significant advantage when complete segmentation of an object is not achievable. For a complicated object shape, we can break the object into simpler sub-shapes and compare sub-shapes. The shape analysis methods discussed in this paper assume that the landmarks are given. We are currently addressing the problem of landmark detection. We believe that the landmarks can be chosen such that a minimum number of landmarks will be necessary to represent all the objects in a library and allow for adequate object recognition. The landmarks should be chosen such that domain specific knowledge about the shapes can be exploited.

The authors would like to thank Professor Fred Bookstein of the University of Michigan for his help and suggestions in preparing this paper.

V. REFERENCES

[1] F. L. Bookstein, *The Measurement of Biological Shape and Shape Change* Springer-Verlag, New York, 1978.

[2] F. L. Bookstein, "Foundations of Morphometrics," *Annual Reviews of Ecology and Systematics,* Vol 13, pp.451-470, 1982.

[3] F. L. Bookstein, "Tensor Biometrics for Changes in Cranial Shape," *Annals of Human Biology,* Vol 11, N0.5, pp.413-437, 1984.

[4] A. I. Borisenko and I. E. Tarapov, *Vector and Tensor Analysis with Applications,* translated by R. A. Silverman, Dover Publication Inc., 1979.

[5] S. A. Dudani, "Aircraft Identification by Moment Invariants," *IEEE Trans. on Computer,* Vol C-26, pp.39-46, Jan. 1977.

[6] D. Gans, *Transformations and Geometries,* Appleton-Century Crofts, 1969.

[7] D. Hilbert, *Foundations of Geometry,* translated by P. Bernay, Open Court, 2nd impression, 1978.

[8] T. Pavlidis, "A Review of Algorithms for Shape Analysis," *Computer Graphics and Image Processing,* Vol 7, pp. 243-258, Apr. 1978.

[9] D. Thompson, *On Growth and Form,* J. T. Bonner ed, Cambridge University Press, 1961.

[10] J. L. Turney, T. N. Mudge, and R. A. Volz, "Recognizing Partially Occluded Objects," *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol PAMI-7, pp.410-421, Jul. 1985.

[11] T. P. Wallace, and O. R. Mitchell, "Analysis of Three-Dimensional Movement Using Fourier Descriptors," *IEEE Trans. Pattern Analysis and Machine Intelligence,* Vol PAMI-2, pp.583-588, Nov. 1980.

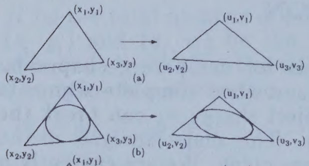

(x1,y1)
(x2,y2)
(x3,y3)
(u1,v1)
(u2,v2)
(u3,v3)
(a)
(b)
(c)
2r
α
β

Fig. 1. Representation of Triangular Deformation. a) given triangles, b) representation of the mapping from inscribed circle to inscribed ellipse, c) mappings of principal dilatations.

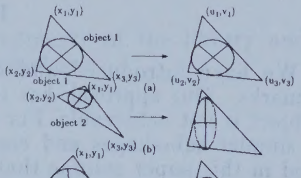

(x1,y1)
object 1
(x2,y2)
(x3,y3)
(a)
object 2
(b)
object 3
(c)
(u1,v1)
(u2,v2)
(u3,v3)
Library Objects
Observed Object

Fig. 2. Classification of objects with three landmarks. The "observed" object is object 1. The shape parameters are:

	s (scale change)	a (anisotropy)
object 1	1.00	1.00
object 2	2.37	2.90
object 3	1.13	1.53

Decision: "observed" object is identified as object 1.

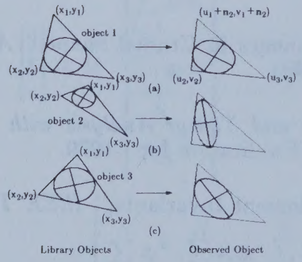

(x1,y1)
object 1
(x2,y2)
(x3,y3)
(a)
object 2
(b)
object 3
(c)
(u1+n2,v1+n2)
(u2,v2)
(u3,v3)
Library Objects
Observed Object

Fig. 3. Classification of objects with three landmarks when the location of one landmark is noisy. The "observed" object is object 1. The noisy landmark is (u_1,v_2), with noise parameter $\gamma=2$. The computed shape parameters:

	s (scale change)	a (anisotropy)
Object 1	0.91	1.15
Object 2	2.17	2.58
Object 3	1.03	1.64

Decision: Observed object is identified as object 1.

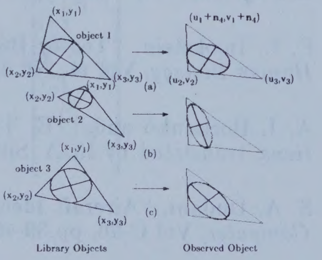

(x1,y1)
object 1
(x2,y2)
(x3,y3)
(a)
object 2
(b)
object 3
(c)
(u1+n4,v1+n4)
(u2,v2)
(u3,v3)
Library Objects
Observed Object

Fig. 4. Classification of objects with three landmarks when the location of one landmark is noisy. The "observed" object is objected. The noisy landmark is (u_1,v_1) with noise parameter $\gamma=4$. The computed shape parameters are:

	s (scale change)	a (anisotropy)
Object 1	0.83	1.34
Object 2	1.96	2.58
Object 3	0.93	1.84

Decision: Observed object is identified as object 1.

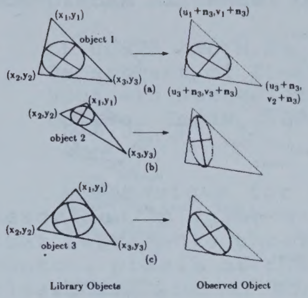

(x1,y1)
object 1
(x2,y2)
(x3,y3)
(a)
(u1+n3,v1+n3)
(u3+n3,v3+n3)
(u3+n3,
v2+n3)
(x1,y1)
(x2,y2)
object 2
(x3,y3)
(b)
(x1,y1)
(x2,y2)
object 3
(x3,y3)
(c)
Library Objects
Observed Object

Fig. 5. Classification of objects with three landmarks when the location of all the landmarks are noisy. The "observed" object is object 1. The noise parameter is $\gamma = 3$. The computed shape parameters are:

	s (scale change)	a (anisotropy)
Object 1	1.01	1.21
Object 2	2.41	2.82
Object 3	1.45	1.51

Decision: Observed object is identified as object 1.

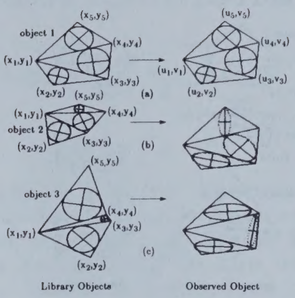

(x5,y5)
object 1
(x4,y4)
(x1,y1)
(x3,y3)
(x2,y2)
(x5,y5)
(a)
(u5,v5)
(u4,v4)
(u1,v1)
(u3,v3)
(u2,v2)
(x1,y1)
object 2
(x4,y4)
(x2,y2)
(x3,y3)
(b)
(x5,y5)
object 3
(x4,y4)
(x3,y3)
(x1,y1)
(c)
(x2,y2)
Library Objects
Observed Object

Fig. 6. Classification of objects with five landmarks. The "observed" object is object 1. The computed shape parameters:

	σ_s	σ_a
Object 1	0.00	0.00
Object 2	1.37	1.30
Object 3	2.24	2.52

Decision: Observed object is identified as object 1.

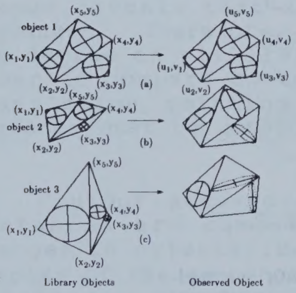

(x5,y5)
object 1
(x4,y4)
(x1,y1)
(x3,y3)
(x2,y2)
(x5,y5)
(a)
(u5,v5)
(u4,v4)
(u1,v1)
(u3,v3)
(u2,v2)
(x1,y1)
object 2
(x4,y4)
(x2,y2)
(x3,y3)
(b)
(x5,y5)
object 3
(x4,y4)
(x3,y3)
(x1,y1)
(c)
(x2,y2)
Library Objects
Observed Object

Fig. 7. Different mode of partitioning the polygon of Fig. 6. The computed shape parameters are:

	σ_s	σ_a
Object 1	0.00	0.00
Object 2	1.37	2.23
Object 3	4.46	5.84

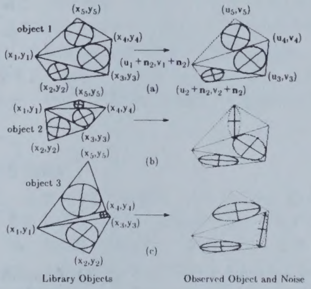

(x5,y5)
object 1
(x4,y4)
(x1,y1)
(x3,y3)
(x2,y2)
(x5,y5)
(a)
(u5,v5)
(u4,v4)
(u1+n2,v1+n2)
(u3,v3)
(u2+n2,v2+n2)
(x1,y1)
object 2
(x4,y4)
(x3,y3)
(x2,y2)
(x5,y5)
(b)
object 3
(x4,y4)
(x3,y3)
(x1,y1)
(c)
(x2,y2)
Library Objects
Observed Object and Noise

Fig. 8. Classification of an object with five landmarks when the location of two landmarks are noisy. The "observed" object is object 1. The noisy landmarks are (u_1, v_1) and (u_2, v_2) with noise parameter $\gamma = 2$. The computed shape parameters are:

	σ_s	σ_a
Object 1	0.13	0.34
Object 2	1.69	1.69
Object 3	2.24	2.99

Decision: Observed object is identified as object 1.

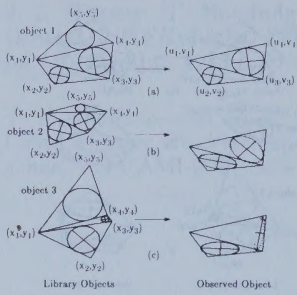

(x_5,y_5)
object 1
(x_4,y_1)
(x_1,y_1)
(x_3,y_3)
(x_2,y_2)
(u_1,v_1)
(u_1,v_1)
(u_3,v_3)
(u_2,v_2)
(a)
(x_5,y_5)
(x_1,y_1)
(x_4,y_1)
object 2
(x_3,y_3)
(x_2,y_2)
(b)
(x_5,y_5)
object 3
(x_4,y_4)
(x_1,y_1)
(x_3,y_3)
(c)
(x_2,y_2)
Library Objects
Observed Object

Fig. 9. Classification of objects with five landmarks when one landmark of the observed object is missing. The "observed" object is object 1. The landmark (u_5,v_5) is missing. The computed shape parameters are:

	σ_s	σ_a
Object 1	0.00	0.00
Object 2	0.49	0.99
Object 3	2.38	3.17

Decision: Observed object is identified as object.

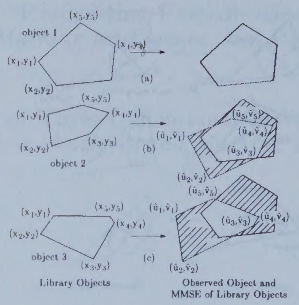

(x_5,y_5)
object 1
(x_4,y_4)
(x_1,y_1)
(x_2,y_2)
(a)
(x_5,y_5)
(x_1,y_1)
(x_4,y_4)
(x_2,y_2)
(x_3,y_3)
object 2
(b)
$(\hat{u}_5,\hat{v}_5)$
$(\hat{u}_4,\hat{v}_4)$
$(\hat{u}_1,\hat{v}_1)$
$(\hat{u}_3,\hat{v}_3)$
$(\hat{u}_2,\hat{v}_2)$
$(\hat{u}_5,\hat{v}_5)$
(x_1,y_1)
(x_5,y_5)
$(\hat{u}_1,\hat{v}_1)$
(x_4,y_4)
$(\hat{u}_3,\hat{v}_3)$
$(\hat{u}_4,\hat{v}_4)$
(x_2,y_2)
object 3
(x_3,y_3)
(c)
$(\hat{u}_2,\hat{v}_2)$
Library Objects
Observed Object and MMSE of Library Objects

Fig. 10. Classification of objects with five landmarks by the MMSE affine transformation method. The "observed" object is object 1.
A_ρ = *shaded area*
Object 1 has $A_\rho = 0$
Decision: Observed object is identified as object 1.

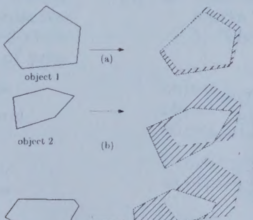

object 1
(a)
object 2
(b)
object 3
(c)
Library Objects
Observed Object and MMSE of Library Objects

Fig. 11. Classification of objects with five landmarks by the MMSE affine transformation method. All five landmarks are noisy with noise parameter $\gamma = 2$. The "observed" object is object 1.
A_ρ = *shaded area*
Object 1 has smalles A_ρ
Decision: Observed object is identified as object 1.

NON-LINEAR ALIGNMENT OF CONTOURS FOR SHAPE CLASSIFICATION

L. GUPTA AND M.D. SRINATH
Department of Electrical Engineering,
Southern Methodist University,
Dallas, Texas 75275.

ABSTRACT

A technique for classifying closed planar shapes is described in which a shape is characterized by a sequence that represents the distance between the centroid and all contour pixels of the shape. Shapes belonging to the same class have similar contour sequences, hence a procedure for classifying shapes is based on the degree of similarity between contour sequences. Dynamic warping which is a non-linear alignment scheme is used to optimally align the sequences in order to compare their similarity. An end point relaxation technique that reduces errors due to inaccurate detection of end points is described and performance results are presented.

1.INTRODUCTION

The contour of a closed planar digitized shape can be described by a sequence $V(i)$, $1 \leqslant i \leqslant N$, in which the samples represent the Euclidean distance between the centroid and all N contour pixels of the shape. An example of a contour sequence is shown in Fig.(2). Since only closed planar shapes are considered, the resulting sequences are circular. A careful examination of sequences of various shapes reveals that similar shapes have similar contour sequences. Therefore, shape similarity can be measured by the degree of simlilarity between contour sequences. In order to compare contour sequences, the effects of shape translation, rotation and scaling on the corresponding sequence must be considered.

2.PREPROCESSING

Contour sequences are shift invariant since pixel distances are computed with respect to the centroid. Changes in orientation result in a circular shift in the samples of the sequence, hence, start point normalization is required. Start point normalization is achieved by selecting the first sample of the sequence as the contour pixel that lies on the major principal axis. Scale changes yield contour sequences differing both in amplitude and duration. Thus, invariance to scaling involves amplitude and length normalization. Amplitudes are normalized by dividing the samples by the standard deviation of the samples. To match the duration of the test shape sequence, the duration of the reference shape sequence is either increased or decreased using linear interpolation.

3.ALIGNMENT OF CONTOUR SEQUENCES

Due to quantization and other geometric deformations in the shape, the contour of a shape is subjected to nonuniform variations. Such variations result in non-linear deviations in segments of the contour sequence. If the normalized contour sequence is regarded as a time sequence, the resulting non-linear deviations in the sequence can be considered as the expansion or compression of certain segments of the time axis. In order to compensate for this non-linear variation in the time axis, dynamic time warping which is a non-linear time alignment scheme is used to optimally align the sequences to measure their similarity. Let the normalized test sequence be T(n), $1 \leq n \leq N$, and let the normalized reference sequence be R(m), $1 \leq m \leq N$. Dynamic alignment finds a mapping w(n) between the time indices n and m to obtain the best match between the two sequences T(n) and R(m). The mapping is subject to a set of end-point and continuity constraints [1-3]. The optimal mapping or alignment path is determined by

$$D = \min_{w(n)} \left[\sum_{n=1}^{N} d(\ T(n), R(w(n))\ ) \right] \tag{1}$$

D is the overall accumulated distance between T(n) and R(m) along w(n). d(t(n),R(w(n))) is the point distance between the n^{th} sample of T and the $w(n)^{th}$ sample of R. The test shape is assigned to the class that yields the smallest accumulated overall distance. Most dynamic time alignment programs assume that the end points of the sequences are precisely known. However, due to noise, the start point as determined by the major principal axis may be incorrect. To anticipate this shift in the start point, an end point relaxation which takes advantage of the circularity of the sequences is used. That is, the reference sequence is circularly extended in both directions by Δ samples to create windows for the true end points of the test sequence to fall within. That is, the extended $(N+2\Delta)$ point reference sequence $R_E(.)$ is

$$\begin{aligned} R_E(.) &= R(N-\Delta), \ldots R(N-1), R(1), R(2), \ldots R(N), R(1), \ldots R(1+\Delta) \\ &= R_E(1), R_E(2), \ldots. R_E(N+2\Delta) \end{aligned} \tag{2}$$

The new end point constraints with respect to the extended reference sequence are

$$\begin{aligned} 1 &\leq w(1) \leq 1+2\Delta \\ N &\leq w(N) \leq N+2\Delta \end{aligned} \tag{3}$$

4.CONCLUSION

Results of testing one hundred noisy shapes from each of the four classes shown in Fig.(1) are shown in the form of a confusion matrix in Table 1. Typical noisy shapes are shown in Fig.(3). The technique performs well on noisy data. Further details and results will be reported in a future paper.

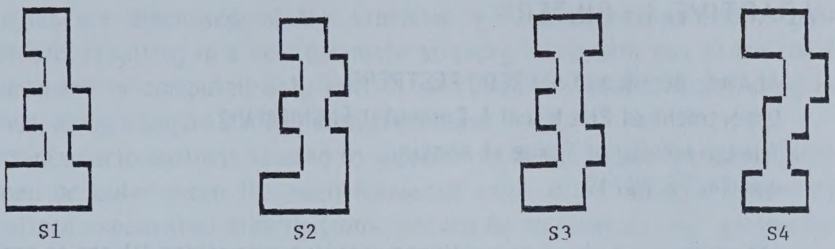


Fig.(1) Reference Shapes used in experiments.

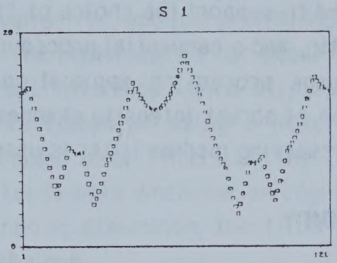


Fig.(2) Contour Sequence of S1.

	S1	S2	S3	S4
S1	100	0	0	0
S2	0	100	0	0
S3	0	0	100	0
S4	0	0	0	100

Table 1.

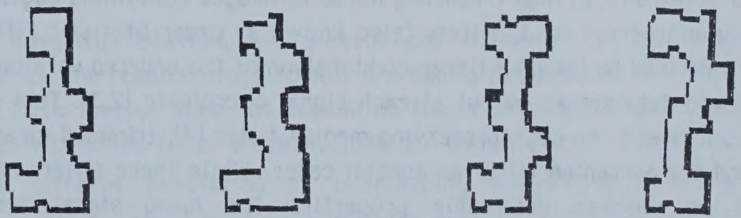


Fig.(3) Typical noisy test shapes

REFERENCES

1. H.SAKOE AND S.CHIBA, "Dynamic programming algorithm optimization for spoken word recognition", IEEE Trans. Acoust., Speech, Signal Process., Feb. 1978, pp.43-49.
2. C.S. MYERS et al, "Performance tradeoffs in dynamic time warping", IEEE Trans. Acoust., Speech, Signal Process., Dec. 1980, pp. 622-635.
3. S.HALTSONEN, "An endpoint relaxation method for dynamic time warping", Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., March 1984, pp.9.8.1-4.

ADAPTIVE L-FILTERS

ALAN C. BOVIK and ALFREDO RESTREPO
Department of Electrical & Computer Engineering
The University of Texas at Austin
Austin, TX 78712

Abstract— A class of nonlinear adaptive smoothing filters is proposed, based on the selection of an appropriate inner or outer trimmed mean filter according to local measurements of the tail behavior (impulsivity) of the noise process. A Monte Carlo analysis is used to support the choice of filter for each measured value of the noise statistic, and a sequential procedure is suggested to reduce computation. For image processing applications, a second statistic is used to mark the location of abrupt intensity changes, or edges; in the vicinity of an edge the trend-preserving median filter is used.

INTRODUCTION

Robust nonlinear filtering techniques are of considerable recent interest as effective alternatives to linear filtering strategies, for applications where there is uncertainty in the noise or signal models, or when there is substantial overlap between the noise spectra and vital portions of the signal spectra (e.g., high frequency noise in images containing edges) [1]. Of particular interest are L-filters (also known as Order Statistic Filters), or filters defined by taking a linear combination of the ordered versions of the windowed data set as output at each signal coordinate [2,3]. This class of filters includes the edge-preserving median filter [4], trimmed mean filters [5], and the averaging filter as special cases. While these filters have been noted to possess desireable properties for many signal processing applications, little theory has been advanced for the analysis and design of these filters for signals that vary in time or space (although there has been some success in characterizing the output space of median filters [4]); presently, hueristicism or nondynamical statistical considerations are used to guide the choice of filter coefficients. In this paper, we present an adaptive approach to the selection of the filter coefficients, where the filter is self-modifying according to local properties of the observed process. At each signal coordinate, the filter output is defined as an appropriately trimmed or outer mean of the windowed data set, according to the measured value of a "selector" statistic, V. The decision as to which filter to use is made by comparing V to a set of thresholds, which span a broad range of noise possibilities characterizing the impulsivity of the noise process. Two strategies are discussed: a) the statistic V is computed at each signal coordinate, resulting in a new estimate at every increment, and b) the local measurement is computed only until a specified acceptance probability is attained, using a sequential decision procedure.

A Monte Carlo analysis is used to support the selection of the appropriate trimmed or outer mean for each measured value of V, using a family of generalized exponential distributions indexed by an "impulsivity" parameter t. The analysis is later extended to include a maximum likelihood approach to the estimation of the density function of V, for use in the sequential procedure. The analysis is done for filter sizes ranging from five to twenty-five, and for several values of the parameter t.

The remainder of the paper is devoted to an extension of the method for image filtering. A two-step approach is suggested whereby an edge map is first computed using a novel edge operator defined as a difference of trimmed mean filters. In the filtering process, the edge-preserving median filter is used whenever an edge is found to pass within the span of the filter window; otherwise, the filter output is defined according to the adaptive technique.

FILTER DESCRIPTION

Some Definitions

To simplify notation, we present the following definitions in one dimension. The results which follow are easily extended to two dimensions, as the filter output does not depend on the relative position of the signal samples within the filter window. Consider then a discrete signal $\{X_j\}$ of iid random samples consisting of overlapping vectors $\{\mathbf{X}_j\}$, where $\mathbf{X}_j = \{X_{j-n}, \ldots, X_j, \ldots, X_{j+n}\}$ is composed of the data values spanned by a filter of length N=2n+1 centered at signal coordinate j. With each vector $\mathbf{X}_j$ we associate the vector of *order statistics* $\mathbf{X}_{(\cdot),j} = \{X_{(1),j}, \ldots, X_{(N),j}\}$ consisting of the algebraically ordered versions of the original data set, so that we have $X_{(1),j} \le X_{(2),j} \le \cdots \le X_{(N),j}$. Given a set of constant valued coefficients $\mathbf{b} = \{b_1, \ldots, b_N\}$, we then define the associated L-filter

$$Y_j(\mathbf{b}) = \sum_{k=1}^{N} b_k X_{(k),j}$$

(Note: for brevity, we will no longer include the positional subscript "j" unless necessary). Estimators of this type have long been recognized by statistics researchers for their optimal properties for estimating the parameters of various distributions; most notably, the sample midrange $(X_{(1)}+X_{(N)})/2$ and the sample median $X_{(n)}$ (recall N=2n+1) are the maximum likelihood estimators of the mean for uniform and double- exponentially distributed variates, respectively, while the sample mean is of course optimal for Gaussian variates.

To effectively filter an additive white noise element obeying an unknown (symmetric) parent distribution, we must concern ourselves with the efficiency of the signal estimator with respect to the tail behavior of the underlying noise process. L-filters can be shown to exhibit good robustness properties over a broad class of noise distributions, but the coefficients must be chosen carefully. For example, the sample median (corresponding to the median filter), which is very efficient for impulsive or heavy-tailed noise, is notoriously inefficient for Gaussian-like noise, while the reverse may be said for the sample mean (running average filter). An alternative that is effective when there is considerable uncertainty in the noise model is the *α-trimmed mean* $M(\alpha)$ (inner mean filter) with coefficients

$$b_k = \begin{cases} \dfrac{1+g-N\alpha}{N(1-2\alpha)} & ;\ k = g+1 \text{ or } N-g \\ \dfrac{1}{N(1-2\alpha)} & ;\ k = g+2,\ldots,N-g-1 \\ 0 & ;\ \text{otherwise} \end{cases}$$

where $N = 2n+1$ and $g = [N\alpha]$ is the largest integer less than $N\alpha$; thus, the filter output is defined as the average of those samples remaining after α percent have been "trimmed" away. Hence, the sample average is given by $M(0)$, and the sample median by $M(1/2)$. Trimmed means effect very reasonable efficiency when the noise distribution has tails heavier than the Gaussian, but are relatively inefficient if the noise is, for example, uniformly distributed (resulting perhaps from signal quantization) [6]. In this case, a more reasonable estimate is the *α-outer mean* (outer mean filter)

$$M^c(\alpha) = [N\cdot M(0) - N(1-2\alpha)M(\alpha)]/2N\alpha,$$

that is, the mean of the observations trimmed away in determining $M(\alpha)$. Outer means are very efficient if the noise distribution has shallower tails than the Gaussian, but are ineffective if the noise is impulsive. For filtering applications involving noisy signals obeying an unknown distribution, it is sensible to consider an adaptive approach that takes advantage of both types of estimates.

Noise Adaptivity

To design an effective adaptive filtering technique, we need an effective measure of the impulsivity of the noise process. Hogg [7] suggested an approach by which the choice of estimator is determined by the sample kurtosis. However, in light of recent studies of the of the kurtosis [8] which suggest that it is undependable, a more reliable statistic has been advanced which is closely related to robust estimation of dispersion [9]. We define

$$V = [U(.05) - L(.05)] / [U(.5) - L(.5)]$$

where $U(\alpha)$ is the average of the largest $N\alpha$ order statistics spanned by the filter window, and $L(\alpha)$ is the average of the $N\alpha$ smallest. As with trimming, fractional items are used if $N\alpha$ is not integer-valued. Clearly, V is a ratio of L-filters; in fact, the numerator is regarded as an excellent estimate of the standard deviation σ for a uniform population, while the term in the denominator is very effective for estimating the σ of a double-exponentially distributed population. We may then define an adaptive filter as follows [9]: at each signal coordinate j, compute V; then for trimmings $\alpha_1 \leq \cdots \leq \alpha_c, \alpha_{c+1} \leq \cdots \leq \alpha_{c+d}$ and appropriately defined thresholds $K_0, \ldots, K_{c+d+1}$, we take the output to be $Y_j = M^c(\alpha_i)$ for $i = 1, \ldots, c$ and $Y_j = M(\alpha_i)$ for $i = c+1, \ldots, c+d$, when $k_{i-1} \leq V < K_i$ (take $K_0 = 0$ and $K_{c+d+1} = \infty$). For the purpose of illustration, however, we will confine ourselves to the simple adaptive estimate

$$Y_j = M = \begin{cases} M^c(1/4) & ; \; 0 \leq V < K_1 \\ M(0) & ; \; K_1 \leq V < K_2 \\ M(1/4) & ; \; K_2 \leq V < K_3 \\ M(1/2) & ; \; K_3 \leq V < \infty, \end{cases}$$

where the coefficients K_i will be defined in what follows.

Monte Carlo Analysis

Consider the family of generalized exponential distributions characterized by the density function

$$g(x \mid t) = C\exp(-\gamma|x|^t) \ ; t \in \mathfrak{R}$$

where $C = (t\gamma^{1/t})/2\Gamma(1/t)$ and $\gamma = [\Gamma(3/t)/\Gamma(1/t)]^{t/2}$. These densities represent a broad range of noise behaviors ranging from very impulsive ($t<1$) to shallow-tailed ($t>2$), approaching the uniform distribution as $t \to \infty$ [2]. For sample sizes N = 5, 9, 13, and 25 we generated 1000 random samples with density g for t = 0.5, 1.0, 1.5, 2.0, 2.5, and 3.0; the corresponding average values of V from this simulation are shown in Fig. 1 below:

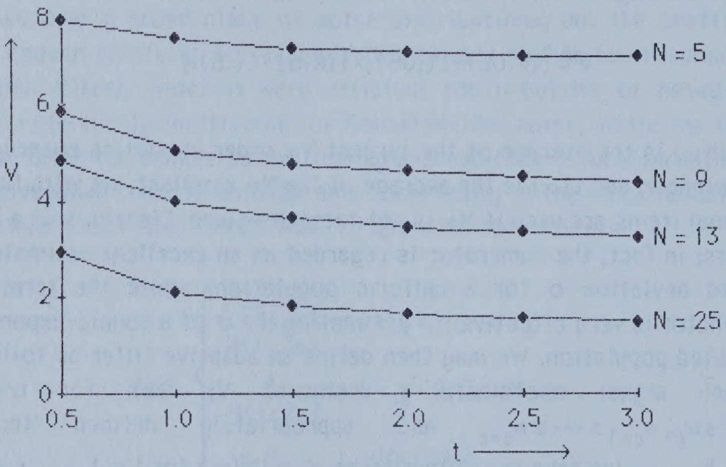


Fig. 1. Emperical values of the selector statistic V as a function of the impulsivity parameter t.

For the simple estimate M, we suggest taking K_1, K_2, and K_3 to be the empirical values of V for t = 2.5, 1.5, and 1.0, respectively, as these roughly correspond to the transitions between shallow, medium, heavy, and very heavy tails in the parent distribution.

A SEQUENTIAL APPROACH TO ADAPTIVE ESTIMATION

A drawback of the adaptive technique described in the preceding is the expense incurred in calculating the selector statistic V, as well as a new estimate at each signal coordinate. If this presents a problem, the sequential procedure we will describe next may be desireable; however we note in passing that, while the noise process may obey a particular noise distribution in the global sense, the observed values may not locally obey this law everywhere in the signal, particularly if a small filter span is used. In this regard, the nonsequential method may be preferable.

Suppose now that we wish to test the multiple hypotheses that the noise process has a first-order density $g(\cdot|t_i)$; $i = 1,\ldots,K$, i.e., has one of K degrees of impulsivity (A more realistic test would involve the composite hypotheses that t lies in one of a set of K intervals. While this is not feasible for our purposes, we may still expect reasonable performance if the t_i are chosen carefully). If we denote by V_j, $j=1,2,\ldots$ the measured values of V from nonoverlapping (hence independent) adjacent filter windows beginning at some initial point in the signal, then a sequential test of the kind developed by Wald [10] is a natural choice. Although there has been no systematic theory developed for dealing with multiple hypotheses, an effective (but suboptimal) test has been suggested as follows [11]. Let

$$\Lambda_{ijm} = \prod_{\ell=1}^{m} \frac{h(x_\ell | t_i)}{h(x_\ell | t_j)}$$

denote the set of likelihood ratios for all pairs t_i,t_j $(i \neq j)$ for m measurements $V_1,\ldots,V_m$, where $h(\cdot|t_i)$ is the conditional density of V given an underlying density $g(\cdot|t_i)$. Further, let A_{ij} $(i \neq j=1,\ldots,K)$ be some $K(k-1)$ positive constants. Then, at the mth stage of the test (m observations), we proceed as follows:

choose t_i if $\Lambda_{jim} < 1/A_{ji}$ for $i \neq j$ $(j=1,\ldots,K)$,

choose t_q if $\Lambda_{qim} > A_{qi}$ for $i \neq q$ $(i=1,\ldots,K)$,

otherwise take an additional observation. Although there is no known optimal criteria (in the Neyman-Pearson sense) for selecting the A_{ij}, we can achieve the upper bound $\Pr\{\text{choose } t_i | t_j \text{ is true}\} \leq \beta/(K-1)$ by taking $A_{ij} = (K-1)/\beta$ for all $i \neq j$. The remaining difficulty involves the estimation of the conditional densities $h(\cdot|t_i)$. As part of our Monte Carlo analysis, we have applied a maximum likelihood approach [12,13] to the estimation of h. Although we lack the space to describe the method here, Fig. 2 shows the density estimates of h for t = 0.5, 1.0, and 2.0 for a window span of N=25. Finally, the sequential procedure is defined as follows: once a given confidence level has been attained for some parameter value $t=t_i$, use the estimate $M(\alpha')$ (or $M^c(\alpha')$) where $\alpha' = \arg\{\min_\alpha\{\mathrm{Var}[M(\alpha)|t_i]\}\}$ to filter the remainder of the signal.

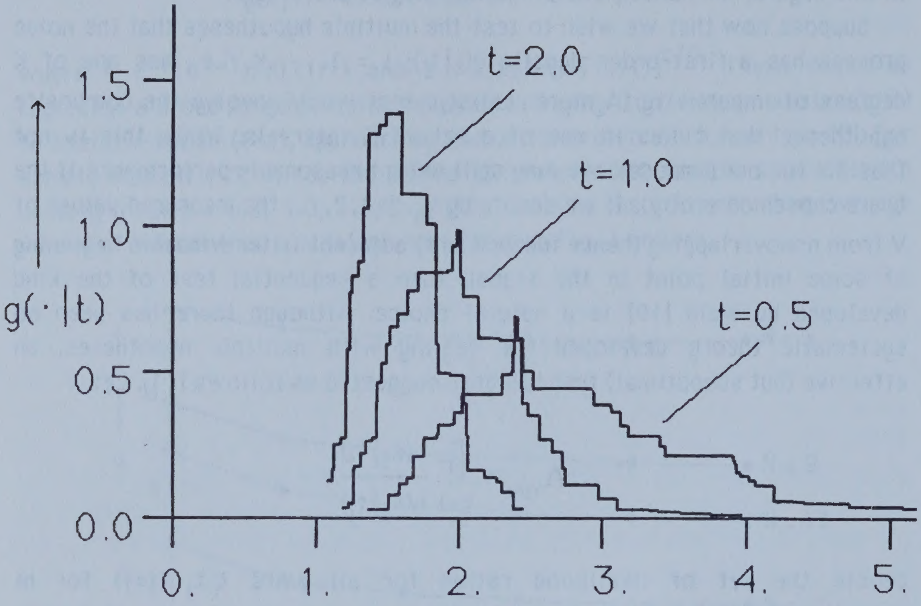


Fig. 2. Emperical conditional density g(·|t) for t=0.5, 1.0, and 2.0, for a filter size of N=25.

EDGE ADAPTIVITY

Thus far we have made no allowance for the time or spatial dependency of the signal. While it may often be reasonable to assume that the signal is relatively smooth within the span, this is not the case for applications such as image filtering, where the edge content should not be disturbed by the filtering process. Plainly, V is not a reliable measure of edge location, but we have found a second statistic defined as a difference of trimmed means, i.e. $Q(\alpha_1,\alpha_2) = M(\alpha_1)-M(\alpha_2)$, to be an effective estimate of edge location; under certain conditions on the signal (roughly, local monotonicity), Q behaves like a difference of low pass filters (which approximates a second order bandpass differential operator [14]), while retaining robustness against impulsive components. Thus, the zero-crossings of Q as applied to the signal are used to assert the presence or absence of edges, which yields an edge map marking the edge contours in the image. In the subsequent filtering process, we decide that an edge lies within the span of the filter if the number of edge points within the window exceeds a threshold. In our preliminary experiments, we have used square (k x k) windows and an edge threshold equal to the span of the filter (k). The following diagram (Fig. 3) illustrates the decision space strategy:

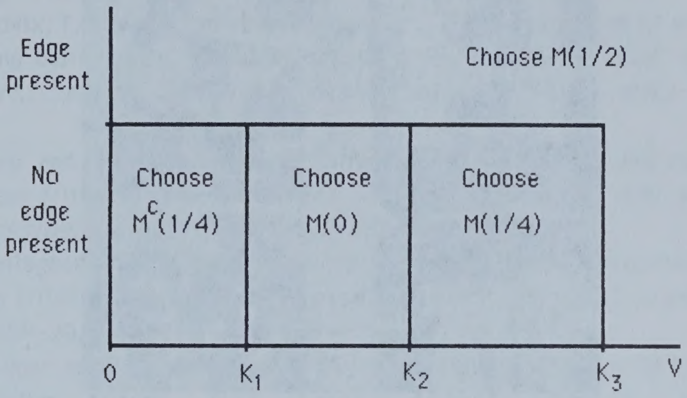
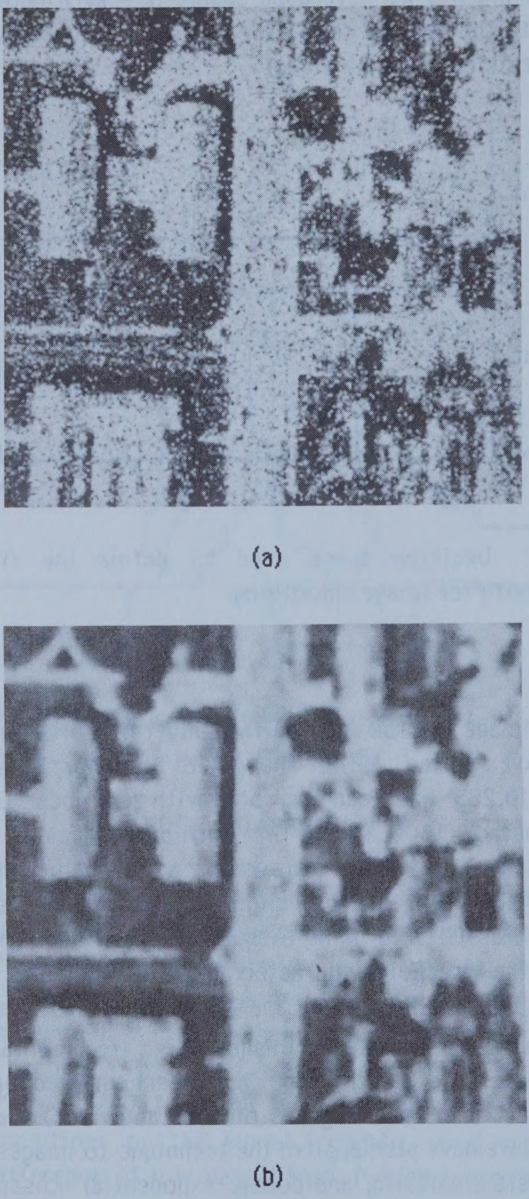


Fig. 3. Decision space used to define the filter adaptivity for image smoothing.

An Example

Consider the image (aerial view of the University of Illinois quadrangle and Matthews Ave., circa 1950) shown in Fig. 4a. This image, with grey-level range [0,255] has been corrupted with zero-mean Gaussian white noise with variance $\sigma^2=1600$, except that each pixel takes on value 0 or 255 with probabilities 0.05; this represents a strongly impulsive noise component departing from the underlying Gaussian process. The cleaned image in Fig. 4b was filtered using a square (5x5) window. For a filter of this size (N=25), the thresholds characterizing the transitions between noise behaviors are $K_1=2.079$, $K_2=1.799$, and $K_3=1.545$. Two observations are worth noting here: first the cleaned image is free from contamination from the impulsive components, and the edges are clearly marked. Second, there is no plataeuing effect (i.e. large regions of constant value) characteristic of the median filter. We have also applied the technique to images contaminated with purely Gaussian, uniform, and double-exponential noises and observed very similar results. Also, we have found that that the method applies very well to images that have been quantized (to as few as 8 grey levels), producing smoothed images with clear edges.



(a)

(b)

Fig. 4. University of Illinois quadrangle image. (a) Gaussian noise with 5% black and 5% white impulses, (b) Cleaned image using 5x5 square adaptive filter.

REFERENCES

[1] S.A. Kassam and H.V. Poor, Robust techniques for signal processing: a survey, *Proc. IEEE,* vol. 73, pp. 433-481, 1985.

[2] A.C. Bovik, T.S. Huang, and D.C. Munson, Jr., A generalization of median filtering using linear combinations of order statistics, *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-31, pp. 1342-1350, 1983.

[3] Y.H. Lee and S.A. Kassam, Generalized median filtering and related nonlinear filtering strategies, *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-33, pp. 672-683, June 1985.

[4] N.C. Gallagher and G.L. Wise, A theoretical analysis of the properties of median filters, *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-29, pp. 1136-1141, 1981.

[5] J.B. Bednar and T.L. Watt, Alpha-trimmed means and their relationship to median filters, *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-32, pp. 145-153, 1984.

[6] J.L. Gastwirth and M.L. Cohen, Small sample behavior of some robust linear estimates of location, *J. Amer. Statist. Assoc.,* vol. 65, pp. 946-973, 1970.

[7] R.V. Hogg, Some observations on robust estimation, *J. Amer. Statist. Assoc.,* vol. 62, pp. 1179-1186, 1967.

[8] R.V. Hogg, More light on the kurtosis and related statistics, *J. Amer. Statist. Assoc.,* vol. 67, pp. 422-424, 1972.

[9] R.V. Hogg, Adaptive robust procedures: a partial review and some suggestions for further applications and theory, *J. Amer. Statist. Assoc.,* vol. 69, pp. 909-923, 1974.

[10] A. Wald, *Sequential Analysis,* New York, Wiley, 1947.

[11] P. Armitage, Sequential analysis with more than two alternative hypotheses, and its relation to discriminant function analysis, *J. Roy. Statist. Soc. Ser. B,* vol. 12, pp. 137-144, 1950.

[12] T. Robertson, On estimating a density which is measureable with respect to a σ-lattice, *Ann. Math. Statist.,* vol. 38, pp. 482-493, 1967.

[13] E.J. Wegman, Maximum likelihood estimation of a unimodal density function, *Ann. Math. Statist.,* vol. 41, pp. 457-471, 1971.

[14] D. Marr and E. Hildreth, Theory of edge detection, *Proc. R. Soc. Lond. B,* vol. 207, pp. 187-217, 1980.

TEXTURE BOUNDARY DETECTION BASED ON LONG CORRELATION MODEL†

R. L. KASHYAP and KIE-BUM EOM
School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

This research deals with the problem of detecting the boundaries between different texture regions in a natural image without assuming any knowledge on the number of regions or kinds of textures. These texture boundaries cannot effectively be detected by traditional edge detection algorithms or pattern classification method because there are many microedges in the same texture and the type and number of textures are not known. An algorithm for detecting these texture boundaries without sacrificing the resolution is proposed in this paper. This algorithm utilizes a *long correlation* texture model with a small number of parameters to characterize a texture. The likelihood principle is used to locate the texture boundary. Empirical studies show that this algorithm successfully detects texture boundaries in the real images when other traditional edge detection algorithms are not successful.

1. INTRODUCTION

If we consider a natural image such as a portrait or outdoor scene, it can be segmented into several regions by texture property. For example, the portrait of a person can be segmented into regions of hair, cheek, background, dress, eyes, etc. Each of these regions are homogeneous in texture property but it does not have uniform intensity and has many micro-edges in it. Unlike human recognition system, any of current algorithms cannot successfully separate these texture boundaries(macro-edges) from micro-edges without sacrificing resolution. The objective of this research is the development of an algorithm which detects boundaries between different texture regions in an image, without sacrificing resolution and without assuming prior knowledge about the number or kinds of textures in the image. This kind of problem is important in image processing and artificial intelligence because a large class of natural images can be considered as a composite of different types of textures.

There are many known edge detection algorithms which are based on the gradient operator (such as Robert, Sobel and Prewitt operators), zero-crossings of smoothed Laplacian operator (Marr and Hildreth, 1980), or directional derivatives method with facet model (Haralick, 1984). However, these traditional edge detection algorithms are not appropriate to detect texture boundaries because there are many micro-edges, edges located inside a texture, which are not eliminated by any of the noise reduction schemes in the traditional edge detection algorithms. The failure of these traditional edge detection methods in detecting texture boundaries arises from a lack of adequate means of characterizing textures.

Thus the popular method in texture segmentation until now is statistical classification method. Even though these approaches were used in many applications, they have many disadvantages. First, the type and number of texture regions in the original image must be known. This is a very unrealistic limitation because the type of textures cannot be known in advance. Second, the standard features for each class should be known. This is also a very unrealistic limitation and sometimes it needs human supervision. Finally, the resolution of the segmented image will be much lower than that of original image. This is because the window size cannot be very small to extract reasonably reliable features and a window is a basic unit in

---

† Partially supported by the office of Naval Research under the grant N00014-82K-0360

segmentation. So, the segmented image is often not acceptable for many applications, such as detecting complex shapes by texture property.

Fractal was introduced by Mandelbrot(1977) and was used to generate three dimensional shapes (relief) which is similar to natural shapes. Recently Pentland(1984) proposed a texture segmentation algorithm using the fractal model in image intensity modelling by assuming that an image intensity generated by fractal relief is also a fractal. However, this assumption is not only theoretically incorrect (Fractal relief generates fractional Gaussian noise image intensity and this never can be a fractal) but also contradicts experimental findings. Peleg et al(1983) reported that "The resulting measures did not support the hypothesis that most textures are fractals." Pentland segmented a picture using the fractal dimension as a feature. The picture is divided into nonoverlapping 8x8 windows and the fractal dimension for each window is estimated by power spectrum method. He claimed that for an image with two textures the histogram of the parameter is bimodal and segmented the picture at the valley of the histogram. Medioni and Yasumoto(1984) applied the same method in segmentation experiment with several real pictures. They reported that the histograms of the estimated parameters for various real pictures are unimodals in many cases and the attempt to change the scale of parameter did not improve the shape of histogram. This result does not coincide with what Pentland claimed. Thus, Medioni and Yasumoto had to find the threshold by a trial and error approach with human decision and it cannot be found by the histogram only as Pentland claimed.

A texture boundary detection algorithm which does not require any prior knowledge about the composition of textures in an image or any human intervention is derived in this research. This algorithm uses the *long correlation* statistical image model(Kashyap and Lapsa, 1984), a generalization of fractional differencing model(Hosking, 1981) and the fractal model(Mandelbrot, 1968; Pentland, 1984) to characterize a texture. This algorithm uses the estimation theoretic approach in detecting boundaries between textures in an image.

2. TWO DIMENSIONAL LONG LORRELATION MODEL

The long term persistency in natural phenomena was first observed by Hurst(1951) in the river flow time series and is explained in detail by Mandelbrot(1967). Mandelbrot and Van Ness(1978) introduced the fractional Brownian motion to model these long term persistencies. This model is used in Pentland's(1984) texture segmentation study and because this is one of the fractals, he called it simply a fractal. Later Granger and Joyeux(1980) and Hosking(1981) introduced another model which has long term dependencies, which is called the fractional differencing model. The above fractional differencing model is generalized into two dimensional case and applied to the generation of textures by Kashyap and Lapsa(1984). This is called the 2-D *long correlation* model.

Suppose that $\{y(i,j)\}$ is a sequence of intensities which follows a two dimensional *long correlation* model and $\{\varsigma(i,j)\}$ is a two dimensional white noise sequence with variance ρ. Then the 2-D long correlation model is represented by the following equation.

$$y(i,j) = (1-z_1^{-1})^{-c}(1-z_2^{-1})^{-d}\varsigma(i,j) \qquad (2.1)$$

$$, \; i=0,\cdots,M-1 \;,\; j=0,\cdots,N-1$$

where z_1^{-1} and z_2^{-1} are unit delay operators in i and j directions respectively.

The differencing parameters c and d characterize the properties of this model along with the noise intensity ρ. The differencing parameters c and d can be estimated in frequency domain by least squares method. Let $\{X(k,l)\}$ and $\{W(k,l)\}$ be DFT's of $\{y(i,j)\}$ and $\{\varsigma(i,j)\}$ obeying (2.1). Then the logarithm of $|X(k,l)|$ is a linear function of parameters c and d.

$$\log|X(k,l)| = -c\log|2\sin(\frac{\pi k}{M})| - d\log|2\sin(\frac{\pi l}{N})| - \alpha + V(k,l) \quad (2.2)$$

$$, \quad k=0,1,\cdots,M-1, \quad l=0,1,\cdots,N-1$$

where $\alpha = \frac{\gamma}{2} - \frac{1}{2}\log\rho MN$, γ is Euler's constant($= 0.5772157$), and $\{V(k,l)\}$ is a white sequence with variance $\frac{\pi^2}{24}$. Define a parameter vector $\theta = [c \; d \; \alpha]^T$. Then the least square estimator $\hat{\theta}$ is obtained by the standard formula.

$$\hat{\theta} = \begin{bmatrix} \hat{c} \\ \hat{d} \\ \hat{\alpha} \end{bmatrix} = \begin{bmatrix} \frac{-12}{MN\pi^2} \sum_{k=1}^{M-1} \sum_{l=0}^{N-1} \log|2\sin(\frac{\pi k}{M})| \log|X(k,l)| \\ \frac{-12}{MN\pi^2} \sum_{k=0}^{M-1} \sum_{l=1}^{N-1} \log|2\sin(\frac{\pi l}{N})| \log|X(k,l)| \\ \frac{-1}{MN} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \log|X(k,l)| \end{bmatrix} \quad (2.3)$$

The asymptotic property of the above least squares estimator was shown in our former research(Kashyap and Eom, 1984). The following theorem 1 gives the asymptotic mean square error of the above estimator.

Theorem 1: The estimators $\hat{c}$, $\hat{d}$ and $\hat{\alpha}$ given by (3.3) are unbiased and consistent in the mean square sense with variances $\frac{1}{MN}$, $\frac{1}{MN}$ and $\frac{\pi^2}{12MN}$ respectively for large M and N.

■

Theorem 1 tells us that the asymptotic mean square errors of estimators are order of $O(\frac{1}{MN})$.

3. MODELS FOR IMAGES WITH MULTIPLE TEXTURES

The idea of detecting edges by maximum likelihood method was first introduced by Kashyap and Bauer(1983). This method is based on the statistical image modelling of textures with a small number of parameters using a quarter plane autoregressive model. Experimental studies(Kashyap 1981, Kashyap and Lapsa 1984) showed that a large class of real textures(such as cork, raffia, calf leather, grass lawn, cotton canvas, wood grain, etc.) can be modelled by one of the statistical image models. Among these image models, the *long correlation* model(Kashyap and Lapsa, 1984) is known for better modelling results of real images with smaller number of parameters than classical image models, such as the Simultaneous Autoregressive model.

Briefly speaking, this boundary detection algorithm based on the estimation of the existence and the location of a boundary in a small size strip of image. The difference of this method with the conventional classification method is the basic unit of this method is pixel not a collection of pixels. Therefore this method gives much finer segmentation result than the results by other methods. The original image which we want to find boundaries between textures is divided into a large number of horizontal and vertical strips and the existence and the location of boundary in each strip is estimated by maximum likelihood method. The final boundaries between textures are obtained by combining these boundaries in all strips.

First the estimation of an edge location in a horizontal strip will be considered. Consider a horizontal strip of image which consists of two kinds of textures with a vertical boundary where the each entry of the array is the pixel intensity. The vertical dimension is assumed small enough to assume vertical boundary and the horizontal dimension is assumed small enough to have no more than one boundary but large enough to have reasonably reliable estimates of parameters. Consider the following

M by N horizontal window which was made up of two kinds of textures as shown below with o indicating one texture and x indicating the other. It is assumed that the vertical boundary between two textures are located at the n^0 th column, n^0 is unknown.

$$\begin{array}{c} n^0 \\ \downarrow \\ \underbrace{\begin{matrix} o&o&o&o&o&o&o \\ o&o&o&o&o&o&o \\ o&o&o&o&o&o&o \\ o&o&o&o&o&o&o \end{matrix}}_{\Omega_1}\underbrace{\begin{matrix} x&x&x&x&x&x&x \\ x&x&x&x&x&x&x \\ x&x&x&x&x&x&x \\ x&x&x&x&x&x&x \end{matrix}}_{\Omega_2} \end{array}$$

Figure 1

Let us define the following symbols for the convenience.

Ω_1 = the set of pixels whose column number $\leq n^0$.
Ω_2 = the set of pixels whose column number $> n^0$.
M_1 = row dimension of $\Omega_1 = M$.
N_1 = column dimension of $\Omega_1 = n^0$.
M_2 = row dimension of $\Omega_2 = M$.
N_2 = column dimension of $\Omega_2 = N - n^0$.

It is also assumed that the textures on the left side of n^0 and right hand side of n^0 follows the following 2-D LC models (3.1) and (3.2) with the parameters $\theta_1 = (c_1, d_1, \alpha_1)^T$ and $\theta_2 = (c_2, d_2, \alpha_2)^T$ respectively.

$$y(i,j) = (1-z_1^{-1})^{-c_1}(1-z_2^{-1})^{-d_1}\varsigma_1(i,j) \ , \quad \text{for}(i,j)\in\Omega_1 \tag{3.1}$$

$$y(i,j) = (1-z_1^{-1})^{-c_2}(1-z_2^{-1})^{-d_2}\varsigma_2(i,j) \ , \quad \text{for}(i,j)\in\Omega_2 \tag{3.2}$$

where $\{\varsigma_i(i,j)\}$ is a 2-D white noise sequence with variance $\frac{1}{M_iN_i}\exp(\gamma - 2\alpha_i)$.

Let us define $\mathbf{X}_1 = \{|X_1(k,l)|\}$ and $\mathbf{X}_2 = \{|X_2(k,l)|\}$ be the modulus of DFT's of $\mathbf{y}_1 = \{y(i,j),(i,j)\in\Omega_1\}$ and $\mathbf{y}_2 = \{y(i,j),(i,j)\in\Omega_2\}$ respectively, $\mathbf{y} = \mathbf{y}_1 \cup \mathbf{y}_2 = \{y(i,j),(i,j)\in\Omega_1\cup\Omega_2\}$ and $\mathbf{X} = \mathbf{X}_1\cup\mathbf{X}_2$. The vector $\mathbf{y}_1$ and $\mathbf{y}_2$ are observations over the left of edge and the right of edge respectively and $\mathbf{y}$ is the whole observations over the entire window. The vectors $\mathbf{X}_1$ and $\mathbf{X}_2$ and $\mathbf{X}$ are transformations of these spatial domain observations to frequency domain. The consistent estimators of θ_1 and θ_2 can be found by the algorithm given in the section 2 from the observations $\mathbf{X}_1$ and $\mathbf{X}_2$. Let us denote these estimators by $\hat{\theta}_1 = (\hat{c}_1, \hat{d}_1, \hat{\alpha}_1)^T$ and $\hat{\theta}_2 = (\hat{c}_2, \hat{d}_2, \hat{\alpha}_2)^T$ respectively.

In the following theorem 2, the log-likelihood function of n^0,$\log p(\mathbf{y}|n^0)$, will be given in terms of parameter estimates $\hat{\theta}_1$ and $\hat{\theta}_2$. The maximum likelihood estimate of the true edge location is the argument which maximize this log-likelihood function by the maximum likelihood principle. Practically, the likelihood functions are computed for all possible edge locations in the window and find the location which maximizes the log-likelihood function.

Theorem 2: Assume that the parameter vector θ_i has the prior density $p(\theta_i)$ which does not vanish in $\mathbf{R}^3$ and has second derivatives in $\mathbf{R}^3$ for all possible n. It is also assumed that θ_1 and θ_2 are independent for all possible n. The asymptotic maximum likelihood estimator of n is the value of n which maximizes the following function J(n), which is computed with the hypothesis of existence of true edge which is located at n.

$$J(n) = \sum_{i=1}^{2}\{\frac{M_iN_i-3}{2}\log(M_iN_i) + M_iN_i\hat{\alpha}_i + \log p(\hat{\boldsymbol{\theta}}_i)\} \tag{3.3}$$

where the symbols used are as defined before.

■

The following theorem 3 gives the maximum likelihood decision rule on the existence of an edge.

Theorem 3: Let J(n) be defined as in theorem 2 and J(0) be the following.

$$J(0) = \frac{MN-3}{2}\log MN + MN\hat{\alpha} + \log p(\hat{\theta}) \tag{3.4}$$

where the estimators of parameters are from the whole observations in the window.

Then the maximum likelihood decision rule for the existence of a texture boundary is the following.

$$\begin{cases} \text{Yes} & , \max J(n) < J(0) + K \\ \text{No} & , \text{otherwise} \end{cases} \tag{3.5}$$

where K is a constant which depends on the prior probability of existence of a boundary in a strip.

■

The boundary detection algorithm is directly followed from the above theorem 2 and theorem 3. It is summarized in the following.

Boundary Detection Algorithm:

1. Divide the image by the small strips(we have used 4x14 windows) in horizontal and vertical directions so that the center portion (the range of the location estimates, we used 5-10th pixels) in long side is not overlapped between different strips in each direction.
2. Estimate the parameters and compute J(n) in (3.3) for each possible edge location(5-10th pixels in our experiment) in a strip. Repeat and find the maximum location n_{max} and the maximum value $J(n_{max})$.
3. Estimate the parameters in the case of no edge and compute J(0) in (3.4). Compare $J(n_{max})$ and J(0) and mark the n_{max}th column(or row) of the corresponding strip in the edge map if the decision rule in (3.5) says yes. If the decision rule in (3.5) gives "no edge" decision, then it is considered that there is no edge in the window.
4. Repeat steps 2-3 until all strips were consumed. After this step, we will have two edge maps, one contains edges in horizontal direction and the other contains edges in vertical direction. These will be called raw edge maps.
5. Cleans the noises in the raw edge maps by edge tracking algorithm which is given in below and combine these two cleaned edge maps and obtain the final detected edges.

4. EXPERIMENTAL RESULTS

Experiment 1: Strips with One or Two Textures

Eight different textures are taken from Brodatz's album(1966). These are calf leather(D24), woolen cloth(D19), beach sand(D29), herringbone weave(D17), raffia(D84), wood grain(D68), straw(D15), and bark of tree(D12) textures. The decision rules derived in section 4 are tested over the various types of texture boundaries which are constructed with the real textures listed above. The following types of 4x14 strips(125 for each type) were constructed:

(1) no texture boundary in the strip
(2) one vertical boundary in the strip
(3) one slanted boundary in the strip with slope 1/2
(4) one slanted boundary in the strip with slope 3/4
(5) one slanted boundary in the strip with slope 1

Typical graphs of decision functions J(n) and J(0) with respect to column numbers for different types of boundaries are given in figure 2. These figures show that the height of peak decreases as the slope of the edge increases but, the peak is at the location which is close to the true boundary location.

Table 1 summarizes the performance of the decision rules over the strips which are constructed with real textures as explained above. This experiment does not include the cleaning procedure given in section 4. In table 1, n^0 and $\hat{n}$ represent the center of the true boundary location(if it has slanted boundary) and the estimated boundary location respectively. Pentland(1984) reported that he achieved 84.4% of correct classification rate with similar textures. This achievement is better than those done by correlation statistics or by co-occurence statistics. As we discussed in section 1, long correlation model which we used in this algorithm is the generalization of fractal model used in Pentland's experiment. Long correlation model has more flexibility in modelling because it has more parameters than fractal model. Thus, the performance of our algorithm is expected to be better. The rate of correct decision($|\hat{n} - n^0| \leq 2$) by our maximum likelihood decision rules is more than 85% over all different kinds of boundaries, which is better than that of Pentland's as expected. In addition, our algorithm does not require any prior knowledge about textures in the image and gives finer segmentation results(we will discuss more on this later).

Table 1. Statistics for Detecting Boundaries With Different Types of Boundaries(sample size = 125)

types of boundaries	no boundary detected	$\lvert n^0-\hat{n} \rvert \leq 2$	$\lvert n^0-\hat{n} \rvert > 2$
no true boundary	87.7%	-	12.8%
vertical	8.8%	88.8%	2.4%
slope=1/2	9.6%	88.8%	1.6%
slope=3/4	10.4%	87.2%	2.4%
slope=1	8.8%	90.4%	0.8%

Experiment 2: Checker Board Image

A 128x128 checker board image(figure 3.a) is constructed by a method similar to Haralick's(1984); however, texture noise is used instead of white noise contamination. Its white checks and black checks are size of 32x32 and are contaminated by normalized wood grain(D68) and grass(D9) textures which have zero mean and variance 250. Figure 3.b illustrates the texture boundaries found by our algorithm with value K in the decision rule for the existence(theorem 3) of boundary 15. The boundary tracking algorithm is not applied in this experiment to compare three different algorithms in the same condition. The detected boundaries correspond to the texture boundaries and each detected texture region does not have micro-edges. Figure 3.c is the result of Marr and Hildreth's algorithm with $\sigma = 5.0$ and a threshold of zero crossing which gives best looking edges. The micro-edges and actual texture boundaries are connected together and cannot be separated even with large σ value. This σ value determines the rate of local averaging. Figure 3.d shows results of Haralick's directional derivative algorithm with a directional derivative threshold of 15 and a second order derivative threshold of 2.0. These thresholds are decided by trial and error in order to give the best looking edges. These edges cannot represent the texture boundaries with any higher level processing. It is obvious that the traditional edge detection algorithms cannot be applied to detect texture boundaries from this experiment.

Experiment 3: Slanted Checker Board Image

To demonstrate the performance of our algorithm on the slanted texture boundary, a 128x128 image(figure 4.a) with repeating diamond patterns contaminated with texture noise. The textures used in generating this image is the same as in the checker board image of experiment 2. The diamond patterns are obtained by rotating 30x30 squares 45 degrees. Figure 4.b is the detected texture boundaries by our algorithm. The detected boundaries in figure 4.b clearly correspond to the texture boundary. Even though the decision rule in the algorithm based on the vertical boundary assumption, this result show that it also works well on the slanted boundaries. Figure 4.c is the detected edges by Marr-Hildreth's algorithm. The threshold value and σ value are the same as experiment 2. This figure shows that this method detects micro-edges as well as macro-edges. Since the micro-edges are not distinguishable from the macro-edges, the segment boundaries are not clear. Figure 4.d is the detected edges by Haralick's algorithm. The threshold values are the same as in the experiment 2. This result shows that this algorithm also does not detect the texture boundary. Figure 4.e is the segmentation result by Pentland's method. Figure 4.f is the histogram of estimated fractal dimension. The threshold is at the valley of the histogram. The window size is the same as in his experiment(8x8). This gives coarse segmentations of the figure 4.a by textures. However because the resolution of this method is too low, the shape of patterns(diamond pattern) does not come out in figure 4.e. Because our algorithm estimates the location of boundary in a window, its resolution is almost one pixel. However, the resolution of the segmented image by Pentland's method decreased by a factor of 8 because 8x8 windows were used and the details less than 8 pixels were not detected.

Experiment 4: Natural Textures

Because many real images can be considered as a composite of different types of real textures, it is necessary to test algorithms on the image have several different textures. Figure 5.a is a 128x128 image constructed by composing five different natural textures with boundaries in different angles. The textures in figure 5.a are cotton canvas, raffia, grass lawn, wood grain, and calf leather in clockwise direction from the left upper corner. Figure 5.b is the detected and cleaned texture boundaries by our algorithm. The edge tracking algorithm in section 4 is used. The detected texture boundaries clearly separates five different regions and these correspond to the macro-edges in figure 5.a. Figure 5.c is the result by Marr-Hildreth's algorithm. The threshold and σ values are the same as in the experiment 2. Figure 5.d is the result by Haralick's algorithm. The threshold values are the same as in the experiment 2. Both figures 5.c and 5.d can detect only the segment boundary in the top left part of image. Other segment boundaries are not obvious.

Experiment 5: Girl's Image

The texture boundary detection algorithm presented in this paper can be used for boundaries between uniform regions as well as texture boundaries. Figure 6.a is a 256x256 image of a girl with flowers in front and blurry background. Figure 6.b is the detected and cleaned texture boundaries. This shows the boundaries between hair, face, flowers, clothes, etc. However, these boundaries does not contain micro-edges in hair and flowers. These results show that the texture boundary detection algorithm performs well on the natural images.

5. DISCUSSIONS AND CONCLUSION

The texture boundaries cannot be detected by any conventional edge detection algorithms as we have seen in section 5. It is because each texture has many micro-edges and the intensity of these micro-edges can be stronger than that of macro-edges. Pattern classification approaches in texture segmentation are not successful either, because they usually require the prior knowledge of the number of classes and give very coarse output. A new algorithm to detect texture boundaries was introduced. It uses the long correlation model to characterize textures and uses the likelihood principle to estimate the texture boundary. There are many advantages of this algorithm over the other conventional methods. This algorithm detects only macro-edges (conventional edge detection algorithms cannot do it) but the resolution of the detected boundaries is almost close to that of the conventional edge detection algorithms (the resolution of the pattern classification approach is much lower).

The rate of correct decision of this decision rules over real textures was over 85% for all different types of boundaries. This achievement is better than that by Pentland's method. If we take account of the quality of segmented image (figure 4), the superiority of our algorithm is obvious. In the experiment with images with slanted boundaries, it was shown that our algorithm also works well over the slanted boundaries.

Through the experiment with a natural image, we have shown that this algorithm can be used for segmenting natural images by texture properties.

REFERENCES

Brodatz, P.(1966), *Textures: A Photographic Album for Artist and Designers,* Dover Publication, New York.

Haralick, R.M. (1984), "Digital Step Edges from Zero Crossing of Second Directional Derivatives," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. PAMI-6, No. 1, January, pp.58-68.

Hosking, J.R.M. (1981), "Fractional Differencing," *Biometrika,* Vol. 68, pp.165-176.

Kashyap, R.L. and Bauer, R.D., (1983), "A Statistical Model Based Approach to Edge Detection in Textures," *Proc. 21st Allerton Conf.,* Chicago, Illinois, October.

Kashyap, R.L. and Lapsa, P.M. (1984), "Synthesis and Estimation of Random Fields Using Long Correlation Models," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. PAMI-6, No.6, November.

Mandelbrot, B.B. and van Ness, J.W. (1968), "Fractional Brownian Motions, Fractional Noises and Applications," *S.I.A.M. Review,* Vol. 10, pp.422-437.

Marr, D. and Hildreth, E. (1980), "Theory of Edge Detection," *Proc. Royal Society of London,* Vol. B.207, pp.187-217.

Medioni, G. and Yasumoto, Y. (1984), "A Note on Using the Fractal Dimension for Segmentation," *Proc. Workshops on Computer Vision,* Annapolis, Maryland, April 30 - May 1, pp.25-30.

Peleg, S. et al.(1983), *Multiple Resolution Texture Analysis and Classification,* vol. CS-TR-1306, University of Maryland, Colledge Park, MD.

Pentland, A.P. (1984), "Fractal Based Description of Natural Scenes," *I.E.E.E. Trans. on Pattern Analysis and Machine Intelligence,* Vol. PAMI-6, No. 6, pp.661-674.

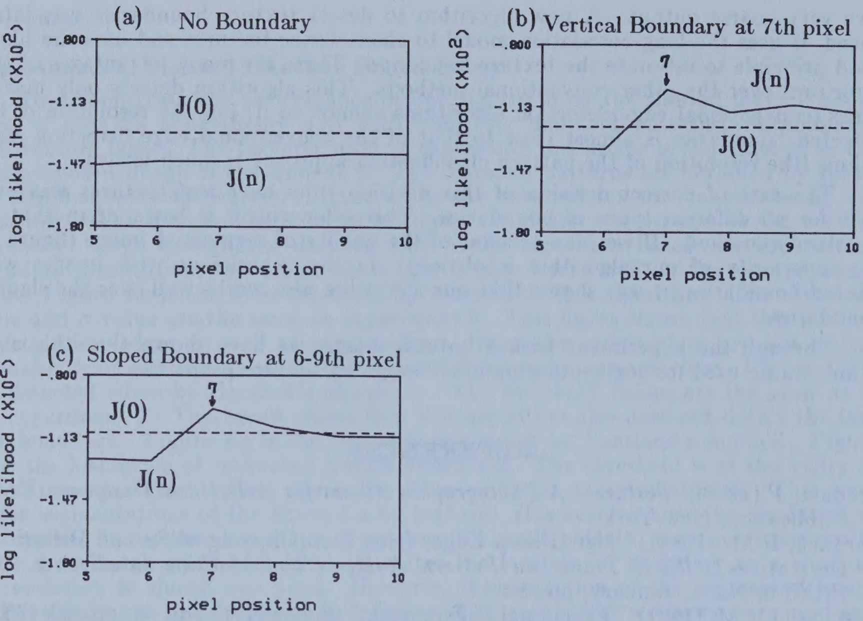


Figure 2: decision functions J(n); solid line, and J(0); dotted line, vs. n. (a) no edge, (b) vertical edge, (c) sloped edge with slope=1

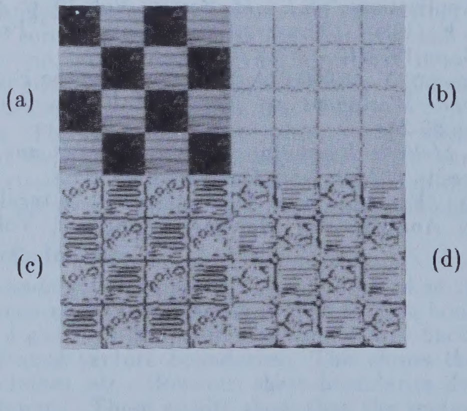
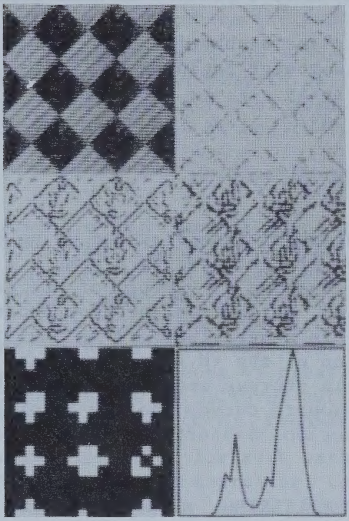
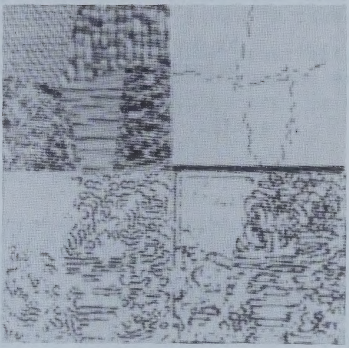
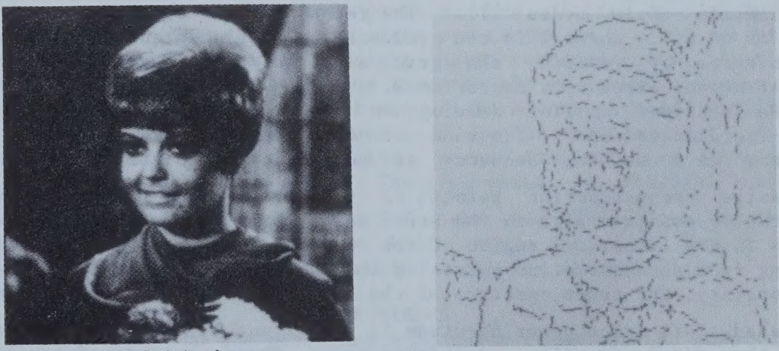


Figure 3: a comparison of our algorithm with traditional edge detection algorithms. (a) original, (b) our algorithm, (c) Marr-Hildreth's, (d) Haralick's





Figures 4-5: comparisons of different algorithms.
(a) original, (b) our algorithm,
(c) Marr-Hildreth's, (d) Haralick's,
(e),(f) Pentland's result and histogram(fig.4)


(a) Original (b) Detected boundaries

Figures 6: result on the natural image.
(a) original image,
(b) detected and cleaned boundaries

TREE SEARCHED CHAIN CODING FOR SUBPIXEL RECONSTRUCTION OF LINE DRAWINGS

by R. Sriraman, Jack Koplowitz and Seshadri Mohan
Department of Electrical and Computer Engineering
Clarkson University
Potsdam, NY, 13676

ABSTRACT

In this paper, we investigate the quantization of line drawings using chain codes. Subpixel accuracy reconstructions are obtained by a simple linear filtering of the chain code points, which achieves a factor of three to four reduction in average reconstruction distortion for smooth curves. Further, the basic chain encoding schemes are generalized to a multipath tree searched encoding scheme. A variation of the (M,L)-algorithm is used to maintain M contending chain code paths in storage and choose the ´best´ path from these. Over a wide variety of source curves, nearly an order of magnitude reduction in average reconstruction distortion over smoothed chain codes is demonstrated.

I. CHAIN CODES AND FILTERING

Line drawings are an important class of pictorial data that occur in diverse applications such as graphs, maps, charts, schematic diagrams, sketches and handwriting. For computer processing, storage or transmission it is necessary to **digitize**, or **quantize**, such line drawings with negligible distortion. Chain codes are line following codes that have been used to quantize such drawings. Chain codes digitize a line drawing, superimposed on a grid of fineness T, as a sequence of contiguous line segments or **links**, l'. In this work we utilize the well known **square quantization chain codes** [1]. The reconstruction error from such chain codes is of O(T), where T is the grid size or pixel distance. Thus doubling the bitrate by reducing the grid size in half reduces the average reconstruction error by a factor of 2.

We consider a line drawing to be a curve described by its (x,y)-coordinates as a function of the distance. We define an error sequence $\{e_i\}$ as the sequence of perpendicular distances from the successive reconstruction points to the original source curve. We also arbitrarily assign a sign to the error terms arising from reconstruction points in the ´interior´ region of the curve and an opposite sign to errors arising from reconstruction points in the ´exterior´ region. If we assume stationarity, the power spectrum of the discrete error sequence $\{e_i\}$ may be estimated. This error power spectrum is approximately white since it arises from the added granular quantizing noise. Chain encoding of bandlimited curves is a highly oversampled encoding scheme, and, the source power is generally concentrated in a narrow band of low frequencies. Suitable low pass filtering of the chain code to attenuate the out of band noise should, therefore, yield improved reconstruction accuracy. The low pass filter should have a bandwidth large enough to transmit the highest frequency components of the source curve with negligible attenuation. The improvement realizable by filtering thus increases inversely with pixel distance or proportionally with the amount of oversampling of the source. We use simple linear FIR filters in our experimental simulations. Experimentally it is seen that such simple linear filtering obtains approximately a factor of three reduction in average reconstruction error for smooth curves. The experimental average reconstruction error for straight lines is close to $O(T^{1.5})$, ccvarying with the slope of the line. Thus doubling the bitrate by reducing the grid size in half reduces the average reconstruction error by approximately a factor of 2.83. However, such linear filtering is not robust, and, can only be applied to smooth low pass source curves.

II. TREE SEARCHED CHAIN CODES

The possible outcomes of any sequential encoder may be graphically listed on a code tree. The chain encoder, like all single path encoders, chooses the locally best path at any instant. In general other paths may exist that are globally superior to the basic chain code. Such paths have higher total noise power than the basic chain code path. The error power spectrum, however, is not flat. These paths have a reduced noise power in the filter passband at the expense of greatly increased out of band noise. This large out of band noise is filtered out by the decoding filter and, hence, a reduction in the total reconstruction noise is obtained. The average reconstruction error for such paths is expected to be of $O(T^2)$.

We utilize tree searching techniques to identify such a globally superior chain code path. Previous applications of tree encoding have attempted to improve the encoding efficiency. Here tree searching is not used to improve the coding efficiency, rather, it is used to exploit the redundancy inherent in the encoded source to reduce the quantization error. The well known (M,L)-algorithm, which performs a sub-optimal search, is used to search the code tree. The algorithm maintains a list of M contending chain code paths of length L and defers a decision on the current symbol to a later time. Each path has associated with it a metric that is an estimate of the fidelity of the path. The metric is computed based on the average filtered reconstruction error and the number of bits used per unit length of the source curve encoded. A tree encoder based on a modified version of the (M,L)-algorithm was tested on a wide variety of source curves including random straight lines, random circles, first order autoregressive curves with continuous tangents and polygonal figures with discontinuous tangents. Excellent results were demonstrated: compared to basic filtered chain codes on the same grid, on the average, an order of magnitude reduction in reconstruction error was obtained at a modest increase in the number of bits used. The performance for straight lines was found to agree very well with the $O(T^2)$ expected. Thus, doubling the bit rate by reducing the grid size in half should yeild a factor of 4 improvement in the average reconstruction error for straight lines.

REFERENCES

1) J. Koplowitz, ´´On the performance of chain codes for the quantization of line drawings,´´ IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-3, No. 2, March 1981.
2) D.L. Neuhoff and K.G. Castor, ´´A rate and distortion analysis of chain codes for line drawings,´´ IEEE Transactions on Information Theory, Vol. IT-31, January 1985.
3) J.B. Anderson and S. Mohan, ´´Sequential coding algorithms : A survey and cost analysis,´´ IEEE Transactions on Communications, Vol. COM-32, No. 2, February 1984.

ITERATIVE RECONSTRUCTION OF BANDLIMITED IMAGES FROM NONUNIFORMLY SPACED SAMPLES *

KEN D. SAUER AND JAN P. ALLEBACH
School of Electrical Engineering
Purdue University
West Lafayette,Indiana 47907

ABSTRACT

We investigate the reconstruction of bandlimited 2-D functions from nonuniformly spaced samples. Three algorithms which iterate between the space and frequency domains are developed. The algorithms differ in the manner in which the known value of the function at sample points is used to correct the reconstruction at each iteration. Convergence of the algorithms is speeded by choosing the largest possible gain for the error correction. The performance of the three methods is compared to that of thin-plate spline interpolation in terms of signal to error energy ratio.

1. INTRODUCTION

Functions must be reconstructed from nonuniformly spaced samples in many scientific and engineering problems. Algorithms have been developed for interpolation of two-dimensional functions by researchers in such diverse areas as mathematics, computer graphics, signal processing, meteorology, and geology. Most interpolation schemes use locally defined polynomial surfaces with function value influenced by each sample inversely in proportion to the distance to the sample. Global polynomial methods are also common, but generally involve the solution of large systems of simultaneous equations and are therefore susceptible to ill-conditioning. A survey of the principles and relative performance of many such methods has been published by Franke[1].

The most desirable reconstruction algorithm is frequently determined by the properties of the function to be recovered. In this paper, we consider the class of two-dimensional functions which are spatially bandlimited. Such functions are important in image processing applications, where lenses and apertures of finite extent normally limit spatial frequency response of imaging systems. We further limit our investigations to functions whose bandregion is circular--a realistic restriction for scenes composed of objects with unknown orientation, and systems of circular lenses.

Iterative reconstruction from partial information has been the subject of a great deal of attention during the past decade, although its mathematical history is much longer. The type of iterative scheme we consider approximates a function f by imposing, at each stage of iteration, constraints on the function which derive from *a priori* knowledge of the function's properties. These constraints are represented as operators P_i, which make some form of correction of the error between f_k, the reconstruction after the k-th iteration, and information known about f. These successively applied operators can be expressed as a composite operator T such that

$$f_{k+1} = Tf_k = P_1P_2...P_nf_k, \tag{1}$$

for n classes of constraints. If the partial information and P_i uniquely specify a function and if the algorithm is convergent,

$$\lim_{k\to\infty} Tf_k = f. \tag{2}$$

At this fixed point of the iteration, Tf = f. If more than one fixed point of T exists, a unique solution may not be possible.

---

* Research supported by the National Science Foundation under Grant ECS-8443337

One of the best known algorithms of this type, originally proposed by Gerchberg and Saxton[2], consists of resubstitution of known data, alternating between a signal and its Fourier transform. Gerchberg and Saxton and, at about the same time, Hirsch **et al**[3] and Gallagher and Liu[4], applied this algorithm to recovery or design of complex-valued signals with prescribed spatial and frequency domain magnitude functions but arbitrary phase functions in both domains. Gerchberg[5] later used the same principle to increase resolution of images known to be of finite duration, but of which only a portion of the spectrum was known. The algorithm extrapolated the spectrum to higher frequencies by alternating between resubstitution of known spectral data and spatial support-limiting. Papoulis[6] applied this method to extrapolation in the space domain of functions whose spectra were known to have finite support. A 2-D adaptation of the Gerchberg algorithm for discrete data was used by Carlotto and Tom[7] for interpolation of surfaces from nonuniformly spaced bathymetric and wind flow data.

Youla[8] showed that the Gerchberg method is a member of a more general class of algorithms which can be described as successive projections of f_k onto linear subspaces of the parent Hilbert space to form f_{k+1}. Each subspace consists of functions constrained by a property of the unknown function. Youla and Webb[9] extended Youla's results to alternating nonlinear projections onto convex sets[†], and used theorems formulated by Opial[10] to show that convergence of the projection method could be speeded significantly via overrelaxation, which extends the projections beyond the borders of the convex sets.

Wiley[11] applied a theorem due to Sandberg[12] concerning convergence of non-linear mappings in Hilbert spaces to the design of an iterative algorithm for reconstruction of bandlimited 1-D functions from non-uniformly spaced interval-average samples. Wiley's method recovers the sampled function without error provided limits on sample spacing and displacement from a uniform sequence are satisfied.

In this paper, we investigate three methods for iterative reconstruction of 2-D images from an unrestricted set of non-uniformly spaced samples. The first is similar to the methods of Wiley and Carlotto and Tom, but is based on alternating projections onto convex sets as described by Youla and Webb. The function f_k is forced first to satisfy known interval-average sample values, and then to satisfy a bandlimiting constraint. To obtain reasonable rates of convergence, overrelaxation is added.

Method 2 employs a triangular tesselation of the image plane for error correction. The first stage of the k-th iteration consists of the construction of an error function e_k which is the triangular interpolation of the error measured at known sample values. The error function is then multiplied by a constant to control stability and the rate of convergence, and added to the most recent approximation f_k of the function. Finally, the result is bandlimited, after which the next iteration begins. Method 3 is identical to Method 2 except for the following modification: The triangularly interpolated error function at every point is weighted by the inverse of the distance to the nearest known sample. More detailed descriptions of all methods and their convergence properties follow in Sec. 2.

In practical applications, truly continuous images are seldom reconstructed. All digital displays and printers have finite resolution, and the methods proposed in this paper would be implemented using a discrete system. The considerations arising in the transition from the continuous to the discrete-parameter case are discussed in Sec. 3. To choose a non-iterative method for comparison to our iterative methods, we tested three interpolation algorithms formulated by Franke and Nielson[13] and Franke[14]: (1) An inverse distance weighted least-squares polynomial interpolant, (2) continuous polynomials defined on triangular segments, and (3) thin-plate splines were compared for mean-square and maximum error in reconstruction of three bandlimited images. The spline method consistently yielded the lowest error according to both criteria. In Sec. 4, the performance of the three iterative methods and thin-plate spline interpolation are compared over a set of 3 images. For each image and each of six different maximum displacements, the methods are compared for 10 sets of samples randomly displaced from a rectangular lattice.

[†] The appendix contains definitions for a few of the terms used in this paper.

2. THREE ITERATIVE ALGORITHMS FOR IMAGE RECONSTRUCTION

2.1 NOTATION

Before discussing our algorithms, we establish notation which we follow for the remainder of the paper. The sampled image which we wish to recover will be represented as $f(\vec{x})$. Here $\vec{x}$ represents Cartesian coordinates in the 2-D image plane; but the algorithms could be applied just as well to functions of any number of dimensions. Random samples of $f(\vec{x})$ are taken at

$$\vec{x}_i \qquad i=1,2,\ldots,N,$$

where N is the total number of available distinct samples. The 2-D Fourier transform of $f(\vec{x})$ is

$$F(\vec{u}) = \int f(\vec{x}) e^{j2\pi\vec{u}\cdot\vec{x}} d\vec{x}.$$

All signals considered have the property

$$F(\vec{u}) = 0, \qquad \vec{u} \notin \Omega, \tag{3}$$

where Ω is a circular bandregion. Each of the algorithms utilizes two operators, P_1^i and P_2, where $i = 1,2,3$ indicates the method to which P_1^i belongs. The bandlimiting operator P_2 is identical for all cases.

2.2. METHOD 1

The first iterative algorithm employs the principle of projection onto convex sets with overrelaxation as proposed by Youla and Webb. In this paper, the convex sets of interest are defined as

$$C_1 = \{g : Sg(\vec{x}) = Sf(\vec{x})\} \tag{4}$$

$$C_2 = \{g : G(\vec{u}) = 0, \quad \vec{u} \notin \Omega\}. \tag{5}$$

The interval-average sampling operator S is given by

$$Sg = \begin{cases} \dfrac{1}{\pi\epsilon^2} \displaystyle\int_{\Psi_i} g(\vec{x}) d\vec{x}, & ||\vec{x} - \vec{x}_i|| < \epsilon,\ i=1,2,\ldots,N, \\ 0, & \text{else}, \end{cases} \tag{6}$$

where Ψ_i is a circular region of radius ϵ centered at $\vec{x}_i$ and

$$0 < \epsilon < \min_{i,j} ||\vec{x}_i - \vec{x}_j||, \quad 1 \le i,j \le N, \quad i \ne j. \tag{7}$$

The operator S realistically represents sampling in imaging systems, where sample values are determined by image intensity integrated over a finite area. We define the algorithm as

$$f_{k+1} = Tf_k = P_2[f_k + \lambda S(f - f_k)], \tag{8}$$

$$P_2 G = \begin{cases} G(\vec{u}), & \vec{u} \in \Omega, \\ 0, & \text{else}. \end{cases} \tag{9}$$

$S(f - f_k)$ is simply the set of measurements of the error between f_k and f at the sample points. The parameter λ is adjusted to control convergence and stability.

Alternatively, we may define the operator P_1^1 as

$$P_1^1 g = \begin{cases} g(\vec{x}) + S[f(\vec{x}) - g(\vec{x})], & \|\vec{x} - \vec{x}_i\| \leq \epsilon, \quad i=1,2,\ldots N, \\ g(\vec{x}), & \text{else}, \end{cases} \tag{10}$$

and express the algorithm as

$$f_{k+1} = Tf_k = P_2[1 + \lambda(P_1^1 - 1)]f_k. \tag{11}$$

P_1^1 and P_2 are projection operators since they clearly map any g to the nearest element of C_1 or C_2, respectively. The degree of overrelaxation of P_1^1 is controlled by λ. Equation (10) is exactly of the form proven by Youla and Webb to converge weakly for any $0 < \lambda < 2$. Therefore, if $C_t = C_1 \cap C_2 \neq \emptyset$, Method 1 converges for $0 < \lambda < 2$. However, as is demonstrated in the experimental section of this paper, it is often desirable to set λ outside the range where convergence is guaranteed.

2.3 METHOD 2

Method 2 also consists of iterative processing by two operators. P_2 is the same as for Method 1. P_1^2 distributes the error correction by triangular interpolation of the error measured at sample points. The algorithm is given by

$$f_{k+1} = Tf_k = P_2[f_k + \lambda P_1^2(f - f_k)], \tag{12}$$

which is of the same form as (8). Here $P_1^2 = RS$ itself is a composite operator, with S as defined above and R representing triangular interpolation among the samples. Let

$$V(\vec{x}) = \{\vec{x}_a, \vec{x}_b, \vec{x}_c\}, \quad 1 \leq a,b,c \leq N, \tag{13}$$

be the vertices of the triangular region of the tesselation that contains $\vec{x}$. Then

$$Rg(\vec{x}) = p(\vec{x};[\vec{x}_a, f(\vec{x}_a)];[\vec{x}_b, f(\vec{x}_b)];[\vec{x}_c, f(\vec{x}_c)]), \tag{14}$$

where $p(\vec{x};[\vec{x}_k, z_k], k = a,b,c)$ is the equation of a plane surface passing through the points $(\vec{x}_k, z_k)$, $k=a,b,c$. The constant λ plays the same role as for Method 1. For the triangular interpolation R, we must create and store a triangular tesselation of the region where f is to be reconstructed. Our triangulation algorithm is based on the method used by Lawson[15] where the max-min angle criterion for adjacent triangles is used to optimize the tesselation.

For a sample set from a uniform rectangular lattice, R is equivalent to convolution with a septahedron. In this case, we have shown [16] using Sandberg's theorem that Method 2 converges for $0.0 < \lambda \leq 0.256$. In the general case where the sample set is not restricted, we have no proof of convergence. In fact, it can easily be shown by counter-example that the algorithm fails the Sandberg and Opial criteria for convergence. However, we find in Sec. 4 that good performance and reasonably rapid convergence are possible with the proper choice of λ.

2.4. METHOD 3

The third algorithm tested is a simple variation on Method 2. In order to reflect the greater uncertainty of our correction by $\lambda RS(f - f_k)$ in areas distant from all known samples, we weight the correction at $\vec{x}$ by a function of

$$\rho = \min_{\vec{x}_j \in V(\vec{x})} \|\vec{x} - \vec{x}_j\|. \tag{15}$$

We define the algorithm as

$$f_{k+1} = Tf_k = P_2[f_k + \lambda P_1^3(f - f_k)], \tag{16}$$

where

$$P_1^3 g = [1 + \alpha\rho]^{-1} RSg. \tag{17}$$

Here α is a parameter that controls the rate at which the weighting falls off with distance to the nearest sample. Method 3 is, in a sense, a compromise between Methods 1 and 2. If we allow α to approach ∞, $[1 + \alpha\rho]^{-1}$ goes to zero, except at sample points, making $P_1^3(f - f_k)$ equivalent to $[P_1^1 - 1]f_k$. Convergence is then proven for 0

$< \lambda < 2$. With α set to 0, P_1^3 is equivalent to P_1^2 and the properties described above for Method 2 apply.

3. CONSIDERATIONS FOR THE DISCRETE CASE

In the discrete case, the reconstructed function at the k-th iteration is defined only at lattice points $\vec{x} = \vec{m}X$ where X is the lattice spacing and $\vec{m} = (m,n)$ denotes an integer pair. For convenience, we let $X = 1$; so the reconstruction is given by $f_k(\vec{m})$. Because the samples at $\vec{x}_i$ are randomly positioned, they will not in general fall on points of the reconstruction lattice. Our measure of reconstruction error is, therefore, necessarily an approximation. We define the error $e(\vec{x}_i)$ at the sample points as the difference between the known sample value at $\vec{x}_i$ and the bilinear interpolation at $\vec{x}_i$ of the reconstructed values at the four enclosing lattice points $(m_i,n_i),(m_i+1,n_i),(m_i,n_i+1)$, and (m_i+1,n_i+1). C_1 is redefined as the set of all functions whose bilinear interpolations at $\vec{x}_i$ equal $Sf(\vec{x}_i)$.

Given this definition of $e(\vec{x}_i)$, Methods 2 and 3 can be implemented as in the continuous case, since the triangulation of the reconstruction plane may be defined for vertices at non-lattice points. However, resubstitution and overrelaxation at sample locations as required for Method 1, is no longer possible. Instead, we distribute the correction $\lambda e_k(\vec{x}_i)$ at $\vec{x}_i = (m_i + \delta_i, n_i + \epsilon_i)$ among the four enclosing lattice points. The operator is

$$P_1^1 = P_{1_N}{}^1 P_{1_{N-1}}{}^1 \cdots P_{1_2}{}^1 P_{1_1}{}^1, \tag{18}$$

where N is the number of samples and

$$P_{1_i}{}^1 g(\vec{m}) = \begin{cases} g(m_i,n_i) + e_k(\vec{x}_i)(1-\delta_i)(1-\epsilon_i), & \vec{m} = (m_i,n_i), \\ g(m_i+1,n_i) + e_k(\vec{x}_i)\delta_i(1-\epsilon_i), & \vec{m} = (m_i+1,n_i), \\ g(m_i,n_i+1) + e_k(\vec{x}_i)(1-\delta_i)\epsilon_i, & \vec{m} = (m_i,n_i+1), \\ g(m_i+1,n_i+1) + e_k(\vec{x}_i)\delta_i\epsilon_i, & \vec{m} = (m_i+1,n_i+1), \\ g(\vec{m}), & \text{else}. \end{cases} \tag{19}$$

For each sample point $\vec{x}_i$, this discrete P_1^1 assigns weighted correction to the four enclosing lattice points in the same proportion as these points contributed to the bilinear interpolation of $f_k(\vec{m})$ at $\vec{x}_i$ for estimation of $e_k(\vec{x}_i)$. The discrete P_1^1 is not a projection, since it will map $f_k(\vec{m})$ onto C_1 only if $\delta_i = \epsilon_i = 0$ for all i. Thus, the Youla-Webb convergence proof does not apply. However, by using Opial's results, we have shown [16] that the algorithm still converges for $0 < \lambda < 2$.

Because we can not, in general, know the exact difference between the original function f and f_k at the sample points, some error in the reconstruction is probable even after the iterations of any method have converged to a fixed point in C_t.

4. EXPERIMENTAL RESULTS

4.1. IMPLEMENTATION OF THE ALGORITHMS

In the discussion of the convergence properties of our three algorithms, there was no restriction on f_0, the input to the first iteration. If the method is convergent, it will be so for any $f_0(\vec{x})$ such that $\int f_0^2(\vec{x})\, d\vec{x} < \infty$. However, the algorithms converge much more rapidly if $f_0(\vec{x})$ is made as close as possible to $f(\vec{x})$. This also decreases the likelihood of divergence if λ is set outside the range in which convergence is assured. We therefore let $f_0(\vec{x}) = RSf(\vec{x})$ for all three methods.

Because these algorithms are computationally intensive (forward and inverse FFT's at every iteration), they will be of greatest practical value if accurate reconstruction is achieved within as small a number of iterations as possible. To further speed convergence, we set λ well above its range of proven convergence and made the algorithms adaptive by monitoring successive changes in f_k as an indicator of convergence or divergence. If $\|f_{k+1} - f_k\|^2 > \sigma \|f_k - f_{k-1}\|^2$, where σ is a constant greater than 1, we judge the iteration to be diverging. We then reset f_k to f_0, and reduce λ by 25%. For the results presented in this paper, σ was 1.1; and λ was initially set to 30.0 for Method 1 and 1.0 for Methods 2 and 3. These values appear to be near or above the limits for probable convergence in the majority of cases. Method 3 was tested using $\alpha = 1$ and 3.

4.2. EXPERIMENTAL IMAGES AND SAMPLE SETS

Three synthetic images were sampled and reconstructed on a 64×64 lattice by each method. This was repeated for each of six ranges of sample displacement using 30 different random sampling patterns, 10 for each image. The images consisted of a simple tetrahedron and the block letters "T" and "O", all of which were circularly bandlimited. The random sampling patterns were generated by adding a sequence of independent, identically distributed random displacements to 81 sample points located on a uniform 9×9 rectangular lattice. Displaced sample points that fell outside the image region were wrapped around. The displacements were uniformly distributed over a range that was allowed to vary from 0 to the dimension of the image region. In the 0 displacement case, these 81 points sampled the images above the Nyquist rate for the circular bandlimit, theoretically allowing exact reconstruction, except for discretization error due to interpolation of reconstructed values at sample points.

4.3. COMPARISON TO SPLINE INTERPOLATION

The three iterative methods were compared to interpolation by thin-plate splines, with a program written by Franke[14]. Spline interpolation makes no use of *a priori* knowledge of a function's properties. For a given data set, it constructs a smooth interpolant with everywhere continuous first derivatives. The interpolant is composed of weighted sums of locally defined functions describing the deformation of thin plates under point loads at sample locations. In our tests, input parameters determining areas of definition of splines and weighting functions were set to Franke's default values.

4.4. RESULTS PRESENTED

All the results which follow represent statistics averaged over 30 cases for each method and sample displacement as described in Sec. 4.2. We measure the quality of a reconstruction by the signal-to-error energy ratio (S/E). Plots in Figs. 1-3 illustrate the performance of spline interpolation and our three iterative methods as functions of iteration number k and range of sample displacement W. W is given in units of Nyquist sampling intervals. In Figures 1 and 2, values at $k = 0$ represent results for f_0, and at $k = 1$, for $f_1 = P_2 f_0$. For the iterative algorithms, we are also interested in the rate of convergence, which we measure by

$$\beta = \frac{\|f_k - f_{k-1}\|^2}{\|f_k\|^2}. \tag{20}$$

In Fig. 4, we plot the convergence rates with W = 0.000, 1.125 and 9.000. The original bandlimited "T" and a set of representative reconstructions for W = 2.25 are illustrated in Figs. 5-10.

4.5. OBSERVATIONS ON RESULTS

The iterative methods performed similarly in all cases. In Figs. 1 and 2, we see that both the mean and standard deviation of the S/E of all three methods increase monotonically with increasing k for W = 2.25. The same behavior was observed for other displacements. Method 2 generated reconstructions with the highest S/E, followed by Method 3 in order of increasing α_2 and Method 1. The difference between the average S/E of Method 2 and that of Method 1 ranged from 0.12 to 3.05 dB depending on W.

The iterative methods exceeded the S/E of spline interpolation after 3 iterations. As illustrated in Fig. 3, the improvement over splines after 40 iterations was between

4.5 and 33.5 dB, depending on W and the iterative method chosen. The quality of the iterative reconstructions degraded rapidly as W exceeded 1.00. This may be due to the fact that with $W < 1.00$, each sample is restricted to a square of unit dimension, preserving the ordering of the original lattice. The standard deviation in S/E was nearly identical for all three iterative methods, ranging from approximately 1 dB at $W=0.5$ to 5 dB at $W=9.0$. The slope of the curves at $k = 40$ in Fig. 1 suggests that significant improvement in the mean S/E might be possible with much larger k. We tested Method 2 for $W = 2.25$ with k up to 512. Mean S/E peaked at 24.59 dB with $k = 402$ and fell to 24.47 dB with $k = 512$. Standard deviation of S/E increased nearly linearly with increasing k for $k > 60$, reaching 10.63 dB with $k = 512$. So, the benefit of large numbers of iterations is limited in this case. The convergence rates illustrated in Fig. 4 are also similar among the iterative methods, decreasing with increasing W.

In Fig. 10, the reconstruction by Method 2 of the bandlimited "T" is visually a very good approximation of the original in Fig. 5. The reconstructions by Method 3 for both values of α were nearly indistinguishable from that in Fig. 10. The output of Method 1 in Fig. 9 has more noticeable error and the thin-plate spline interpolation in Fig. 7 is clearly poorer than either of the iterative reconstructions at $k=10$. The benefit of incorporating our knowledge of the function's bandlimitedness is apparent in the restoration by the iterative algorithms of the "T" in the vicinity of the coordinates (15,40) and (22,10), which can be seen in Fig. 6 to be sparsely sampled.

5. CONCLUSION

We have developed three iterative methods for reconstruction of bandlimited images from non-uniformly spaced samples. Each algorithm alternates application of frequency and space domain constraints at each iteration, but the methods differ in their mode of error correction in the space domain. Method 1 compensates only at sample points, while Methods 2 and 3 employ triangular interpolation of error across the entire image region. For Method 3, the interpolated error function at any reconstruction point is weighted inversely with increasing distance to the nearest vertex of the enclosing triangle. Our algorithms seek to maximize convergence rates by making the error correction at each iteration as large as possible. The performance of the three methods was tested on a set of three images with varying ranges of sample displacement. Experimental results indicate that the iterative schemes produce signal to error ratios which increase monotonically for at least several hundred iterations, and which, within a few iterations, exceed those of thin-plate spline interpolation. After 40 iterations, the iterative algorithms result in substantial improvement over spline interpolation.

6. APPENDIX: DEFINITIONS

Convex Set

A set C is defined to be convex if for any f, g $\in$ C and $0 \leq \mu \leq 1$, $\mu f + (1 - \mu) g \in C$.

Non-expansive Mapping

Mapping $T : C \rightarrow C$ is non-expansive if $|| Tf - Tg || \leq || f - g ||$ for all f, g $\in$ C.

Asymptotically Regular Mapping

Mapping $T : C \rightarrow C$ is asymptotically regular if for all $g_0 \in C$, the sequence $g_{k+1} = Tg_k$ satisfies $|| Tg_k - Tg_{k-1} || \rightarrow 0$ as $k \rightarrow \infty$.

ACKNOWLEDGEMENT

The authors wish to gratefully acknowledge the assistance of R. W. Franke in supplying programs for polynomial and spline interpolation.

REFERENCES

[1] R. W. Franke, "Scattered Data Interpolation-Tests of Some Methods", *Math. of Comput.*, Vol. 38, pp. 181-200, Jan 1982.

[2] R. W. Gerchberg and W. O. Saxton, "A Practical Algorithm for the Determination of Phase from Image and Diffraction Plane Pictures", *Optik*, Vol. 35, pp. 237-246, April 1972.

[3] P. M. Hirsch, J. A. Jordan, and L. B. Lesem, U. S. Patent 3,619,022, Nov. 1972.

[4] N. C. Gallagher and B. Liu, "Method for Computing Kinoforms that Reduces Reconstruction Error", *Applied Optics*, Vol. 12, pp. 2328-2335 , Oct. 1973.

[5] R. W. Gerchberg, "Super-resolution through Error Energy Reduction", *Optica Acta*, Vol. 21, pp. 709-720, Sept. 1974.

[6] M. J. Carlotto and V. T. Tom, "Practical Interpolation of 2-D Surfaces Using the Gerchberg Algorithm," *Digest - Topical Meeting on Signal Recovery and Synthesis with Incomplete Information and Partial Constraints*, Incline Village, NV, 12-14 January 1983, pp. (WA3-1)-(WA3-4).

[7] A. Papoulis, "A New Algorithm in Spectral Analysis and Band-Limited Extrapolation", *IEEE Trans. Circuits Syst.*, Vol. CAS-22, pp. 735-742, Sept. 1975.

[8] D. C. Youla, "Generalized Image Restoration by the Method of Alternating Orthogonal Projections", *IEEE Trans. Circuits Syst.*, Vol. CAS-25, pp. 694-702, Sept. 1978.

[9] D. C. Youla and H. Webb, "Image Restoration by the Method of Convex Projections-Part 1", *IEEE Trans. Med. Imaging*, Vol. MI-1, pp. 81-94, Oct. 1982.

[10] Z. Opial, "Weak Convergence of the Sequence of Successive Approximations for Nonexpansive Mappings", *Bull. Amer. Math. Soc.*, Vol. 73, pp. 591-597, July 1967.

[11] R. G. Wiley, "Recovery of Bandlimited Signals from Unequally Spaced Samples", *IEEE Trans. Comm.*, Vol. COM-26, pp. 135-137, Jan. 1978.

[12] I. W. Sandberg, "On the Properties of Some Systems that Distort Signals-I", *Bell. Syst. Tech. Journal*, Vol. 43, pp. 2033-2046, Sept. 1963.

[13] R. W. Franke and G. Nielson, "Smooth Interpolation of Large Sets of Scattered Data", *Int. Journal Numer. Meth. Eng.*, Vol. 15, pp. 1691-1704, Nov. 1980.

[14] R. W. Franke, "Smooth Interpolation of Scattered Data by Local Thin Plate Splines", *Comp. & Math. with Appl.*, Vol. 8, No.4, pp. 273-281, 1982.

[15] C. L. Lawson, "Software for C^1 Surface Interpolation", *Mathematical Software Vol. III*, J. Rice, ed., pp. 161-194, Academic Press, New York, 1977.

[16] K. D. Sauer, "Iterative Reconstruction of Bandlimited Signals from Nonuniformly Spaced Samples," M.S. Thesis, Purdue University, Dec. 1985.

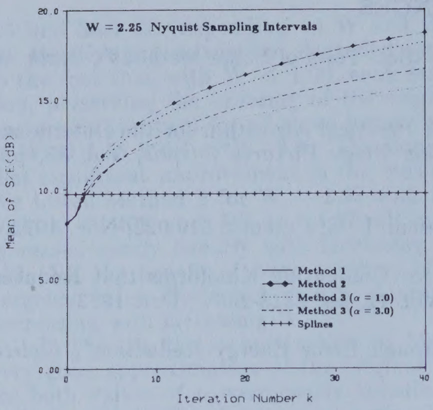


Fig. 1. Mean signal to error energy ratio for 30 cases as a function of iteration number.

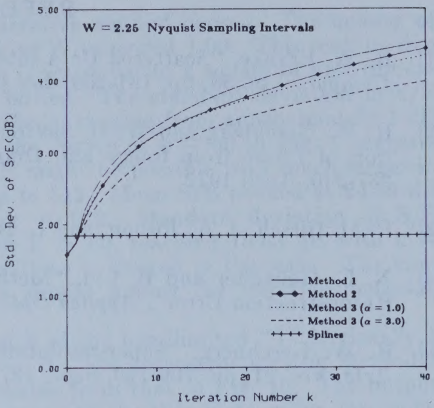


Fig. 2. Standard deviation of signal to error energy ratio as a function of iteration number.

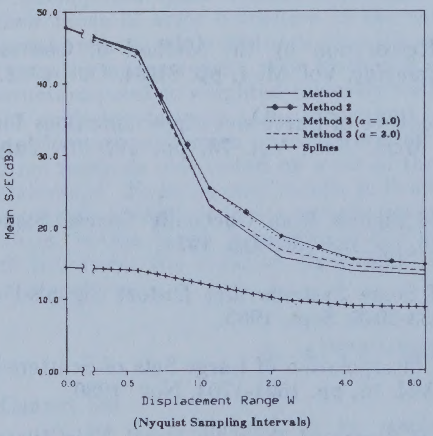


Fig. 3. Mean signal to error energy ratio for 30 cases as a function of range of sample displacement.

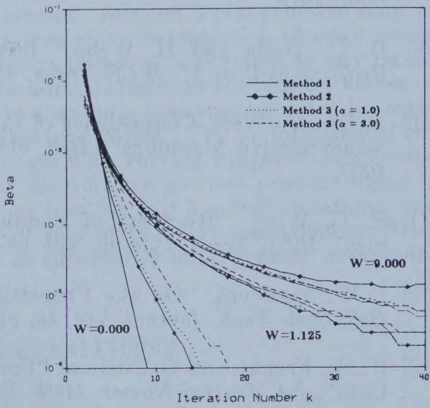


Fig. 4. Normalized energy in the difference between reconstructions at successive iterations for 30 cases as a function of iteration number.

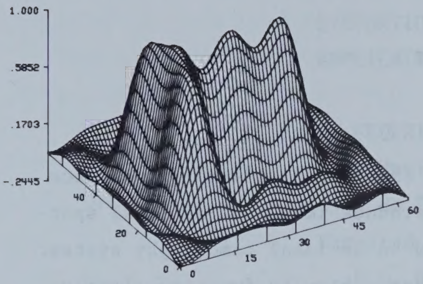


Fig. 5. Block letter "T" bandlimited to a circle of radius 4.5 cycles/image dimension.

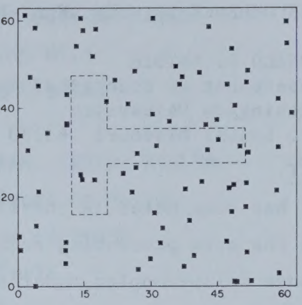


Fig. 6. Sample pattern with W = 2.25.

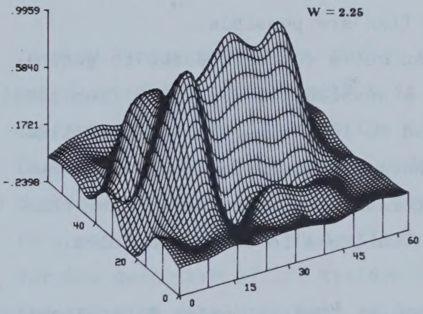


Fig. 7. Thin-plate spline reconstruction of bandlimited "T". S/E = 11.33 dB.

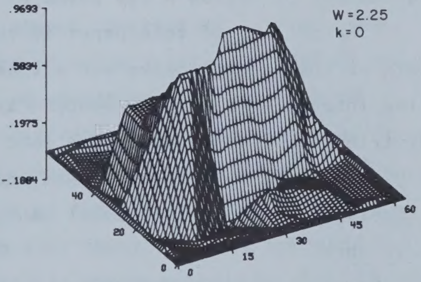


Fig. 8. Triangular interpolation of bandlimited "T". S/E=8.88 dB. W = 2.25.

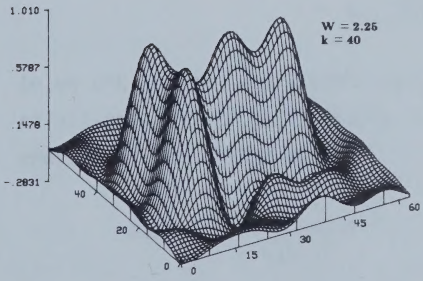


Fig. 9. Reconstruction of bandlimited "T" by Method 1. S/E = 15.72 dB.

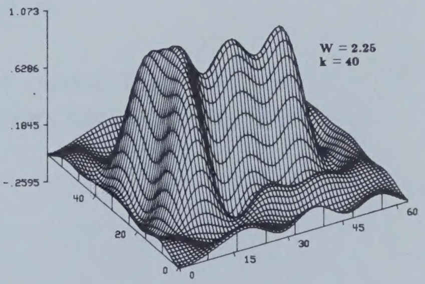


Fig. 10. Reconstruction of bandlimited "T" by Method 2. S/E = 18.97 dB.

A CORRESPONDENCE BETWEEN DELAY-DOPPLER RADAR-IMAGING AND RADIONUCLIDE IMAGING

DONALD L. SNYDER
Department of Electrical Engineering
Washington University
St. Louis, Missouri 63130

ABSTRACT

It has been noted in the literature that there is a correspondence between the data processing required to produce images in an ideal spotlight-mode delay-doppler radar system and in an ideal tomography system. This correspondence presumes an ideal radar ambiguity-function closely approximating an impulsive fence, without sidelobes; this is an approximation that is not well met in practical radar systems. It also presumes a tomography system that produces ideal line integrals through the medium being imaged; this is a good approximation for x-ray systems where the equivalent of narrow x-ray beams of high flux are possible.

The purpose of this paper is to point out a correspondence to another form of tomography system where there is a substantial departure from ideal line integrals (1); these systems are used to form images of radionuclide distributions. The processing that has been developed for these systems may suggest ways to improve delay-doppler radar images by permitting the effects of nonideal, practical ambiguity functions to be accommodated.

Reference

1. Donald L. Snyder and Harper J. Whitehouse, "Delay-Doppler Radar-Imaging Using Chirp-Rate Modulation," Proc. Dixieme Colloque sur le Traitement du Signal et ses Applications, GRETSI, Nice, France, May 1985.

RATE DISTORTION BOUNDS FOR QUOTIENT BASED DISTORTION FUNCTIONS WITH APPLICATION TO LPC SYSTEMS

ANDRES BUZO, FEDERICO KUHLMANN, CARLOS RIVERA
Universidad Nacional Autónoma de México
División de Estudios de Posgrado
Facultad de Ingeniería
P.O. Box 70-256
04510 México, D.F.
MEXICO

ABSTRACT

In many linear predictive coding (LPC) speech compression systems, the encoding of the LPC parameters is performed using a product codebook scheme. One approach is to organize the set of reproduction LPC models as the Cartesian product of a vector codebook describing the shape of each reproduction LPC model and a scalar codebook describing the gain or energy. In this paper, we obtain theoretical bounds of the rate distortion function for the gain term of LPC systems (which used about 10 to 25 percent of the transmission rate), as well as an asymptotic approximation to the performance of the optimal scalar quantizer of these gain terms, when the overall fidelity criterion is the Itakura-Saito distortion measure. These approximations and bounds are compared with experimental results.

I. INTRODUCTION

In an LPC speech compression system the speech frames are characterized by an all-pole filter described by its z-transform $\sigma/A(z)$ (or simply σ/A), where σ is the filter gain,

$$A(z) = 1 + \sum_{k=1}^{m} a_k z^{-k},$$

and the excitation signal is either a periodic pulse train with period τ, zero time-average mean and unit time-average energy for voiced sounds (where τ is equal to the pitch value), or a zero-mean, unit variance sequence of independent random variables for unvoiced sounds. The set of parameters $a_1, \ldots, a_m$, σ, τ that define this model have to be extracted from the sampled speech signal, quantized and transmitted to the receiver. The rate of such parameter extraction is usually on the order of 50-100 Hz to follow the time varying overall characteristics of the input speech signal. At the receiver, speech can then be synthesized using the above all-pole model and the voiced-unvoiced information.

From rate distortion theory [1] it is known that, in order to achieve a low rate at a given distortion, it is better to quantize the whole set of parameters $a_1, \ldots, a_m$, σ, τ as a single vector or block. Since there does not exist a meaningful distortion measure for LPC systems that explicitly uses the pitch τ, this last quantity is usually encoded separately, which results in a suboptimal compression system with a product code scheme. Coding of the excitation parameter is therefore not considered here. In [2], Gray et-al proposed a system that quantizes a vector which includes the parameters a and σ of the above model, using the modified Itakura-Saito distortion measure [3] as the fidelity criterion. Even though this distortion measure is not universally accepted, it is used explicitly or implicitly in LPC systems and it has the form

$$d_{IS}(|\sigma/A|^2, |\hat{\sigma}/\hat{A}|^2) = \frac{\alpha}{\hat{\sigma}^2} - \ln\frac{\sigma^2}{\hat{\sigma}^2} - 1 \qquad (1)$$

where

$$\alpha = \int_{-\pi}^{\pi} |\hat{A}\,(e^{j\theta})|^2 \left|\frac{\sigma}{A(e^{j\theta})}\right|^2 \frac{d\theta}{2\pi}\ , \qquad (2)$$

σ/A is the all-pole model to be quantized and $\hat{\sigma}/\hat{A}$ is its quantized version.

A suboptimal system which allows relatively large rates to be simulated with reasonably small memory requirements was proposed in [4], using the same distortion measure as the one in [2] and (1) but with a different expression obtained by adding and subtracting $\ln \alpha$ to the right term of (1), that is,

$$\left(\ln\frac{\alpha}{\sigma^2}\right)+\left(\frac{\alpha}{\hat{\sigma}^2} - \ln\frac{\alpha}{\hat{\sigma}^2} - 1\right). \qquad (3)$$

This expression reflects the so-called "gain separation" property of the Itakura-Saito distortion measure. The first summand in (3) does not depend on $\hat{\sigma}$, while the second is nonnegative and equals zero if and only if $\hat{\sigma}^2 = \alpha$. Thus the first term is the gain-optimized distortion (i.e. Itakura distor tion) and is the value of the distortion when σ/A is encoded as $\sqrt{\alpha}/\hat{A}$. The second term is the contribution to the distortion when $\sqrt{\alpha}$ is quantized as $\hat{\sigma}$. Thus, appropiately defining d' and d", both of the distortion measures (1) and (3) have the form

$$d_{IS}\,(|\sigma/A|^2,\ |\hat{\sigma}/\hat{A}|^2) = d'(|\sigma/A|^2,\ |1/\hat{A}|^2) + d''(\sigma^2_{opt},\ \hat{\sigma}^2), \qquad (4)$$

where

$$\sigma_{opt} = \sqrt{\alpha}. \qquad (5)$$

Using this "gain separation" property, a scheme to encode the LPC parameters is presented in [4]. It basically consists of encoding the normalized all-pole model 1/A, using d' and computing the value of the optimal gain $\sqrt{\alpha}$. Then the optimal gain $\sqrt{\alpha}$ is encoded with distortion d". We will refer to d" as the *scalar Itakura-Saito distortion*.

Algorithms to design the quantizers of the all pole model and the gain term using d' and d" respectively are given in [4].

A further simplification was proposed later for designing the normalized all-pole model quantizer. It consists of using $\frac{\alpha}{\sigma^2} - 1$ (called the gain-normalized Itakura-Saito distortion) instead of $\ell n \frac{\alpha}{\sigma^2}$ (called the gain-optimized distortion). The approach was successful, in the sense that a system based on vector quantization with 10 bits per frame for the normalized filter 1/A, scalar quantization with 5 bits per frame for the optimal gain, 5 bits per frame for the pitch (at 50 models per second this yields 50 (10+5+5)=1000 bps) resulted in quantitative and subjective quality close to that of an LPC system with approximately optimal scalar quantization and optimal bit allocation at a rate of 2150 bps.

Thus, it can be seen that the gain plays an important role in these encoding systems [4] (25 percent of the transmission rate). Furthermore, when the parameters of the normalized all pole-model (or a transformation of them) are scalar-quantized, then the value of the optimal gain, using the Itakura-Saito distortion as fidelity criterion after they are quantized, is given by (5), and the fidelity criterion that should be used to design a quantizer for the optimal gain is the second term of the right hand side of (4). The gain usually requires about 10% of the transmission rate in these systems.

Another possible application of quotient based distortions can be found in the encoding of the excitation signal of LPC systems. Since the excitation signal s_n for voiced sounds is a nondeterministic periodic pulse train, $(s_n = s_n + \tau)$ for a particular speech frame it can be represented in terms of its Fourier series as

$$s_n = \sum_{k=0}^{t-1} a_k \cos \frac{2\pi k}{t} n + b_k \sin \frac{2\pi k}{t} n \ .$$

Encoding this waveform can be done using a variety of fidelity criteria, and in particular, it can be done with a criterion closely related to the one used for encoding the LPC models, that is, a discrete version of the Itakura-Saito distortion [7]:

$$d_D = \frac{1}{2t} \sum_{k=0}^{t-1} \left\{ \frac{S_k}{\hat{S}_k} - \ln \frac{S_k}{\hat{S}_k} - 1 \right\},$$

where $S_k = E\left\{ a_k^2 + b_k^2 \right\}$. This measure, as well as its continuous version, both are based on cross-entropy concepts [2,7].

In this paper we present bounds for the rate-distortion function for quotient based distortion measures, and in particular for bandwidth compression systems that use the scalar Itakura-Saito distortion as fidelity criterion. In addition, an asymptotic approximation to the performance of the optimal scalar quantizer of these gain terms is derived. Finally, we use these bounds to evaluate the performance of an LPC gain quantizer in a product-codebook environment as described above, when the overal encoder performance is determined by means of the Itakura-Saito distortion function.

II.BOUNDS TO THE RATE-DISTORTION FUNCTION FOR THE SCALAR ITAKURA-SAITO DISTORTION

A. Quotient based distortion measures.

Let the n-dimensional vector $\underset{\sim}{x} = (x_1, \ldots, x_n)$ be the output of a continuos amplitude, discrete time source. Assume that the components of $\underset{\sim}{x}$ are independent, identically distributed (i.i.d) random variables with absolutely continuous probability distribution function and density function P(x). Let $\underset{\sim}{y} = (y_1, \ldots, y_n)$ be an n-dimensional vector which is a member of the reproduction alphabet. The distortion that results from reproducing $\underset{\sim}{x}$ as $\underset{\sim}{y}$ is given by $d_n(\underset{\sim}{x},\underset{\sim}{y})= \frac{1}{n} \sum_{k=1}^{n} d(x_k,y_k)$ where $\{d_n, 1 \leq n < \infty\}$ is the single letter fidelity criterion generated by the nonnegative function d. We know that the rate-distortion function R(D)[1] is a tool to determine the least rate at which information about the source must be conveyed to the user in order to achieve a prescribed fidelity as measured by the average value of a given distortion function. In particular, for a quotient type distortion measure, $d(x,y)= \rho(x/y)$ and a positive source $(P(x)=0,\ x<0)$, from Berger's results [1] and [10] it follows that:

Proposition 1.

For a positive, discrete memoryless source, using a quotient based single letter fidelity criterion, the rate distortion function R(D) is parametrically lower bounded by

$$R(D_s) \geq R_L(D_s)=h(P)-\int_0^\infty dx\ P(x)\ell n\ x - h(F_s) + \int_0^\infty dxF_s(x)\ell n\ x\ , \tag{7}$$

$$D_s= \int_0^\infty dxF_s(x)\rho(x)\ , \tag{8}$$

where h(P) is the differential entropy of the probability density $P(\cdot)$, $F_s(x)$ is a probability density function defined by

$$F_s(x)= \frac{1}{x}\ e^{-s\rho(x)} \left(\int_0^\infty \frac{dz}{z}\ e^{-s\rho(z)}\right)^{-1} \tag{9}$$

and s is a positive real parameter. In addition, for any positive value of this parameter, the lower bound $R_L(D)$ (called the Shannon lower bound), coincides with R(D) (i.e., (7) is a equality) if and only if the random variables x associated with the source output can be represented as the product of two independent random variables, one of which is distributed according to the probability density function $F_s(x)$, as given by (9). This in turn is equivalent to the fact that the integral equation

$$P(x) = \int_0^\infty \frac{dy}{y}\ F_s\left(\frac{x}{y}\right)\ Q_s(y) \tag{10}$$

has a solution $Q_s(y)$ which is a valid density function.

Proposition 2.

For a nonnegative discrete memoryless source under a quotient single-letter distortion measure, the rate distortion function is upper bounded by

$$R(D_s) \leq R_u(D_s)=h(Q_s)-\int_0^\infty dxQ_s(x)\ell n\ x - h(F_s)+\int_0^\infty dxF_s(x)\ell n\ x, \tag{11}$$

where

$$Q_s(y) = \int_0^\infty dx\; q_s(y|x)\; P(x) \tag{12}$$

$$q_s(y|x) = F_s(x/y)\, x/y^2, \tag{13}$$

and D_s is given in (8).

We now turn to the particular case in which the single-letter fidelity criterion is the scalar version of the Itakura-Saito distortion measure.

B.1 Scalar version of the Itakura-Saito measure.

Let the scalar Itakura-Saito single letter distortion measure between source and reproduction symbols be given as in the right hand component d" in (4), namely by,

$$d_{IS}(x,y) = x/y - \ell n\; x/y - 1, \tag{14}$$

and let

$$\rho(z) = z - \ell n\; z - 1 \tag{15}$$

Substituting (15) into (9), it can be easily concluded that for the scalar Itakura-Saito measure and for any source distribution, the density $F_s(x)$ (of the multiplicative noise see Berger,[1,pp.94]) is the gamma density with unit mean and second moment given by $E(x^2) = (s+1)/s$, that is

$$F_s(x) = \frac{s^s}{\Gamma(s)}\, x^{s-1}\, e^{-sx}\;, \tag{16}$$

where $\Gamma(s)$ is the gamma function.

Using this density in the last two terms of the right side of (7) yields

$$h(F_s) = s + \ell n\; \Gamma(s) - \ell n\; s - (s-1)\;\psi(s) \tag{17}$$

$$\int_0^\infty dx F_s(x)\, \ell n\; x = \psi(s) - \ell n\; s = -D_s \tag{18}$$

where D_s is given by (8) and $\psi(s)$ is the psi function. Therefore, upon substituting (15)-(17) into (7) and (8), the parametric equations for the lower bound of the rate distortion function result in

$$R_L(D_s) = h(P) - E(\ell n x) - (s - s\psi(s) + \ell n\; \Gamma(s)) \tag{19}$$

$$D_s = \ell n\; s - \psi(s). \tag{20}$$

It can be observed from (19) that this lower bound is, as expected, a function of the source output probability density function $P(\cdot)$. We now turn to the analysis of particular cases.

B.2 Gamma source.

As it will be seen later, the gamma distribution plays a similar role in the context of the scalar Itakura-Saito distortion measure, as does a Gaussian distribution in the context of mean-squared-error encoding. Thus, assume that the density function P(x) of the discrete memoryless source is of the gamma family with parameters (α,β), that is

$$P(x) = \frac{\beta^{-\alpha}\, x^{\alpha-1} e^{-x/\beta}}{\Gamma(\alpha)}, \quad x \geq 0, \tag{21}$$

with positive parameters α and β .

Thus, (19) can be reduced to the form

$$R_L(D_s) = (\alpha - s) + \ell n \frac{\Gamma(\alpha)}{\Gamma(s)} - \Big(\alpha\ \psi(\alpha) - s\ \psi(s)\Big)\ . \tag{22}$$

Let D_{max} be the minimum average distortion at zero transmission rate. Then $D_{max} = \min\limits_y E\ \{d_{IS}(x,y)\} = E\{d_{IS}(x, E\{x\})\} = - E\{\ell nx\} + \ell n\ E\{x\}$, and for the gamma source this yields

$$D_{max} = \ell n\ \alpha - \psi(\alpha). \tag{23}$$

An interesting property of gamma sources using d_{IS} as the single letter fidelity criterion, and the lower bound defined parametrically by means of (22) and (20) is given in the following proposition (all the proofs are given in [10]).

Proposition 3.

The Shannon lower bound $R_L(D)$ to the rate distortion function of a discrete memoryless gamma source with parameters α and β (see (21)) under the scalar Itakura-Saito distortion measure, coincides with R(D) on the interval $0 \leq D \leq D_{max}$. This rate distortion function will be denoted by $R_{G,\alpha}(D)$.

B.3 General source distribution.

Consider now the case where the discrete memoryless source has an unknown distribution, and only the mean value and the maximum distortion at zero rate is known. The following propositions give the greatest Shannon lower bound and an upper bound for the rate distortion function.

Proposition 4.

Let Λ be the family of positive discrete memoryless sources with mean value equal to a and minimum average distortion at zero rate equal to D_{max}, i.e., $-E\{\ell n\ x\} + \ell n\ E\{x\} = D_{max}$.

The source that achieves the greatest Shannon lower bound of the rate distortion function has a marginal density function which is a gamma function with parameters (α,β), where α is given by the solution of the equation $\ell n\ \alpha - \psi(\alpha) = D_{max}$, and $\beta = a/\alpha$. That is, $R_{G,\alpha}(D) = \max_{P\varepsilon\Lambda} R_L(Ds)$.

Proposition 5.

Let a positive discrete memoryless source be such that the mean value of the output and the minimum average distortion at zero rate are given by $E\{x\}=a$, $-E\{\ell n x\} + \ell n\ E\{x\} = D_{max}$.

Then the rate distortion function is upper bounded as

$$R(D_s) < R_u(D_s) = (r-s) + \ell n \frac{\Gamma(r)}{\Gamma(s)} - (r\psi(r) - s\psi(s)),\ D_s = \ell n\, s - \psi(s) \qquad (24)$$

for all $s > 1$ or equivalently, for distortions $D_s < C = 0.57721 \ldots$ (Euler's constant), and the parameter r is the solution to $\ln r - \psi(r) = D_{max} - D_s + \ell n \frac{s}{s-1}$. Furthermore, when $D_s \to 0$, then $R_u(D_s) \to R_{G,\alpha}(D_s)$. It can be concluded that if $\tilde{R}(D)$ is the rate distortion function with respect to the scalar Itakura-Saito measure, of the worst source with minimum average distortion at zero rate equal to D_{max}, then, $R_{G,\alpha}(D) \leq \tilde{R}(D) \leq \bar{R}_u(D)$, and so, asymptotically, *the gamma source is the worst case to be encoded with the scalar Itakura-Saito measure, when only the minimum average distortion at zero rate is known.*

In the next section the bounds that were derived in the previous paragraphs are used to evaluate the performance of Lloyd's scalar quantization of a memoryless gamma source when the distortion is measured by the scalar Itakura-Saito function. A better evaluation is one based on lower bounds derived as asymptotic performances of optimal scalar quantizers as $N \to \infty$. These results are based on Berger's derivations [2, ch. 5] and [8].

B.4 Asymptotically optimal quantization

Given a fixed N, an optimal quantizer minimizes a distortion function over the N reproduction points and over the boundaries between the levels. Gish and Pierce [9] derived asymptotic approximations for the distortion, as N is large and when the distortion is a function of the difference between source and reproduction symbols. This was later generalized for other distortions and to several dimensions. In particular, we follow [8] to obtain an asymptotic lower bound for the quantizer which minimizes the scalar

Itakura-Saito distortion. Let a discrete-time memoryless source generate positive, i.i.d. random variables with density $P(x)$ whose support is (a,b). Let T_k be the thresholds of an N level scalar quantizer $(a=T_o, b=T_N)$. If the single letter distortion from reproducing x by y is of the quotient type, $\rho(\frac{x}{y})$, then the average distortion is

$$D = E\ \{\rho(x/Q(x))\} = \sum_{k=1}^{N} \int_{T_{k-1}}^{T_k} P(x)\ \rho(\hat{x}/x_k)\ dx. \tag{25}$$

If N is large and $\rho(\frac{x}{y})$ is the scalar Itakura-Saito function, then the average distortion can be bounded as shown in the next proposition.

Proposition 6.

Under the above set of hypotheses, and if $P(x)$ is gamma with parameters (α,β), then [10]:

$$\tilde{D} \geq \frac{1}{\Gamma(\alpha)} \sum_{k=1}^{\infty} \frac{1}{(2k+1)!} \ \frac{1}{(2N)^{2k}} (2k+1)^{\alpha} \left[\Gamma\left(\frac{\alpha}{2k+1}\right)\right]^{2k+1} \tag{26}$$

III.EXPERIMENTAL EVALUATION

Two sets of eight scalar quantizers were designed with Lloyd's Algorithms [10] : one was based on a training sequence of gain terms, generated from 11000 frames of pre-processed sampled male speech. The training sequence of optimal gains was obtained from 11000 sets of LPC coefficentes, quantizing the normalized all-pole model coefficients using [4] and computing the gain for each frame using (5). D_{max} was estimated with arithmetic means. Since our results are interesting when applied to a gamma source (the LPC gains are not gamma), a second set of scalar quantizers was designed, using a training sequence of 25000 samples from a memoryless gamma source, with D_{max} as computed from the LPC optimal gains of the first training sequence. We also compute the bounds for $R_1(D)$ of a source with a fixed but unknown distribution (we omit the subindex 1). From the experimental D_{max}, using proposition 4 and 5, we compute the worst case Shannon lower bound $R_{G,\alpha}(D)$ and the upper bound $R_u(D)$ of (24). The functions $R_{G,\alpha}(D)$, $R_u(D)$ are presented in Fig.1, from which it can be seen that the worst case rate distortion function for the optimal LPC gains source is practically known in the range $D<0.1$ or $R>2.5$ bits/gain term $(R_u(D)-R_{G,\alpha}(D)<0.06$ for $D<0.1)$. The worst case rate distortion function is within 2% of the rate distortion function for the above memoryless gamma source, $R_{G,\alpha}(D)$. The performance of Lloyd's quantizer for the source which generates the optimal LPC gains, inside as well as outside the training sequence is also shown. The quantizer performance is within 3% of the worst-case source with the same D_{max} (from 1 to 8 bits/sample). Fig. 2 shows the rate distortion function for the gamma source with scalar Itakura-Saito function, a lower bound to the asymptotic performance of the optimal scalar quantizer, and the performance of the scalar quantizer designed with [5,6]. Note that for a fixed distortion, the best scalar quantizer is at least one bit/sample above the rate distortion bound for large rates. For lower rates (5 bits/sample) the performance of the designed quantizer and the lower bound to the asymptotic approximation coincide, (but the validity of the asymptotic approximation is questionable in this region). It can be seen that scalar quantizing a gamma source at 5 bits/sample, a reduction of 1.3 bits/sample is in principle possible for the same D.

IV. CONCLUSIONS

Some information theoretic results, previously applied to difference distortions can be extended to measures based on quotients. The Itakura-Saito distortion, used in speech coding [2,3] , has a scalar version which is quotient based, and shares the above properties with the original Itakura-Saito function. The performance of Lloyd's quantizer (scalar) for a source of optimal

LPC gains using the scalar Itakura-Saito distortion, is comparable to the rate distortion function, with the same fidelity criterion, of the worst-case source (gamma) with the zero-rate minimum distortion (for R>2 bits/symbol). The worst-case source for the scalar Itakura-Saito distortion and a given D_{max} is gamma.

REFERENCES

[1] T. Berger, Rate Distortion Theory: A Mathematical Basis for Data Compression, Prentice Hall, Englewood Cliffs, NJ, 1971. [2] R.M. Gray, A.H. Gray, Jr., G. Rebolledo, and J.E. Shore, "Rate distortion speech coding with a minimum discrimination information distortion measure", IEEE Trans. Inform. Theory Vol. IT-12, pp.708-721, Nov. 1981.[3] R.M. Gray, A. Buzo, A.H. Gray and Y. Matsuyama, "Distortion Measures for Speech Processing", IEEE Trans. on Acoust., Speech, and Signal Processing, Vol.ASSP-28, pp.367-376, Aug.1980. [4]M.J. Sabin and R.M. Gray, "Product Code Vector Quantizers for Waveform and Voice Coding", IEEE Trans. Acoust., Speech and Signal Processing, Vol. ASSP-32, pp.479-488, June 1984. [5] S.P. Lloyd, "Least Squares Quantization in PCM", IEEE Trans.Inform.Theory Vol.IT-28, pp.129-137, March 1982. [6] Y. Linde, A. Buzo and R.M. Gray, "An algorithm for vector quantizer design", IEEE Trans.Commun., Vol.COM-28, pp.89-95, Jan.1980. [7] J.E. Shore, "Minimum Cross-Entropy Spectral Analysis", IEEE Trans.Acoust., Speech and Signal Processing, Vol.ASSP-29, pp.230-237, April 1981. [8] R.M.Gray and A.H.Gray, Jr., "Asymptotically Optimal Quantizers". IEEE Trans.Inform.Theory Vol.IT-23 , pp.143-144, Jan.1977. [9] H. Gish and J.N. Pierce, "Asymptotically Efficient Quantizers", IEEE Trans.Inform.Theory, Vol.IT-14, pp.678-683, Sept. 1968. [10] A. Buzo, F. Kuhlmann, C. Rivera, "Rate distortion bounds for quotient based distortions with application to Itakura-Saito distortion", IEEE Trans. Inform. Theory, to appear.

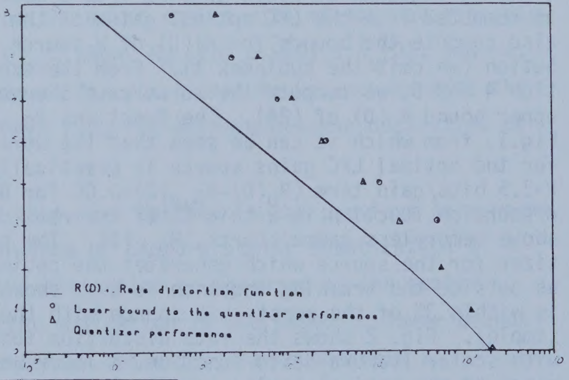


Fig. 1
Rate (bits/sample) vs. Distortion ($\log_{10}$(average scalar Itakura-Saito)

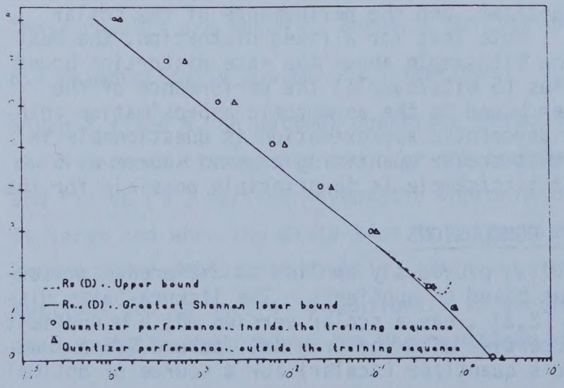


Fig. 2
Rate (bits/sample) vs. Distortion ($\log_{10}$ (average scalar Itakura-Saito)

FAST VECTOR QUANTIZERS

NADER MOAYERI, DAVID L. NEUHOFF, and WAYNE E. STARK
Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, Michigan 48109

Abstract

This paper presents a technique for speeding up the encoding process for vector quantization. We assume the codebook (i.e., the set of quantization vectors) of some vector quantizer is given and we seek a fast quantization rule for this codebook. This is achieved by breaking the encoding into two stages; first, a source vector is mapped to the closest quantization vector of a "fine" codebook for which there is a fast quantization algorithm (fine means its quantization vectors are very closely spaced). Second, a table look-up finds the closest (in distortion) quantization vector of the original codebook to that of the fine codebook. We present the results of applying this approach to a number of codebooks designed for various sources (*iid* Gaussian, first-order Gauss-Markov, and samples of speech). Our approach offers a substantial decrease in the complexity at the cost of a small increase in distortion and a sizeable increase in storage.

Introduction

It is well known that under great generality (nearly) optimum performance in the rate/distortion sense can be obtained using vector quantizers, and that there are algorithms for design of such[c.f.1]. However, one problem with this source coding approach is the frequent need for large blocklengths, which translates into very large numbers of quantization vectors. This potentially makes the encoding process computationally expensive, because a brute force implementation of the quantization rule would require each source vector to be compared with all quantization vectors in order to find the "best" one.

A vector quantizer consists of a codebook C and a quantization rule Q, and will be denoted here by an ordered pair (C,Q). The codebook C is a set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\}$ of M points in the k-dimensional Euclidean space R^k. The quantization rule Q is characterized by a partition $S=\{S_1, S_2, \ldots, S_M\}$ of R^k such that $Q(\mathbf{x})=\mathbf{x}_i$ if $\mathbf{x} \epsilon S_i$; $i=1,2,\ldots,M$. The performance of the quantizer is determined by its rate $R=\frac{1}{k}\log_2 M$ and by its mean squared error $D=\frac{1}{k} E\,||\mathbf{X}-Q(\mathbf{X})||^2$, where $||\mathbf{x}-\mathbf{y}||$ denotes the Euclidean distance between $\mathbf{x}$ and $\mathbf{y}$. We often write $D(Q)$ or $D(S)$ instead of simply D. The minimum distortion quantization rule Q^* for a codebook C is governed by the partition S^* with $S_i^*=\{\mathbf{x}: ||\mathbf{x}-\mathbf{x}_i|| \le ||\mathbf{x}-\mathbf{x}_j||, j\neq i\}$. Let $D^*(C)$ denote its average distortion.

In evaluating the performance of a vector quantizer, the average distortion and rate are not the only important parameters. The complexity of the algorithm that implements the quantization rule is important, also. For example, the brute force algorithm for implementing the minimum distortion quantization rule requires storing the M quantization vectors (codewords) and computing distortion between a given source vector and all the M quantization vectors in the codebook C. To measure the complexity of any algorithm we will count the number of arithmetic operations required per source sample, as a measure of computational complexity, and the total size of storage for the algorithm, as a measure of storage complexity. Specifically, we count the number of additions/subtractions, the number of multiplications/divisions, and the storage in bytes (4 bytes per floating point number).

Fast Vector Quantization

Given a codebook C we wish to find a quantization rule Q having an algorithm that is computationally inexpensive (i.e., fast) and has distortion $D(Q)$ close to $D^*(C)$. Our approach is the following. We first choose a very fine quantizer (C', Q') for which there is a fast quantization algorithm (fine means that $D(Q') << D^*(C)$). To each quantization vector of the fine codebook C', let us associate the closest quantization vector of the original codebook C, and let us construct a table which stores this association. Formally, this association is represented by a map ϕ from C' into C given by $\phi(\mathbf{c}') = Q^*(\mathbf{c}')$. The table will have as many entries as there are quantization vectors in the fine codebook. The fast quantization rule Q encodes a vector $\mathbf{x}$ of source samples by first finding the closest quantization vector $\mathbf{c}'$ of C', then performing a table look-up to determine the quantization vector $\mathbf{c}$ from the original code C associated with $\mathbf{c}'$, i.e., $Q(\mathbf{x}) = \phi(Q'(\mathbf{x}))$.

Since $Q(\mathbf{x})$ is not necessarily the closest quantization vector of C to $\mathbf{x}$, it follows that $D(Q) \geq D^*(C)$. The increase in distortion, $D(Q) - D^*(C)$, depends strongly on the number of quantization vectors in C', and hence the size of the table. The larger the table, the smaller the resulting increase in distortion over $D^*(C)$. The computational complexity of the algorithm described above for quantization rule Q equals that of the algorithm for Q' plus that of a table look-up (negligible). We assume that each entry of the look-up table requires $\log_2 M$ bits of storage. The storage required for the algorithm for Q is that of the algorithm for Q' plus that of the table. In some cases the latter is dominant. Therefore, if the algorithm for Q' has low computational complexity, then the computational complexity of Q will be significantly reduced over the brute force algorithm for Q^*. The price is the increased storage of Q' and the look-up table and some increase in average distortion. This approach is in some respects similar to the approach of [2].

There are a number of codes that can be used as the auxiliary fine vector quantizer. Recall that such codes are expected to have simple quantization algorithms. One such code is the tree-searched vector quantizer (TSVQ)[3]. A binary TSVQ (C, Q) with blocklength k and rate R is characterized by a binary tree of depth $L = kR$, each node of which is labelled by some vector in R^k called a decision vector. The quantization vectors are the 2^L terminal nodes. The quantization rule is generated by the following algorithm: First $\mathbf{x}$ is compared to the two decision vectors at depth 1. We move to the node corresponding to the closer decision vector. We then compare $\mathbf{x}$ to the decision vectors at the two nodes stemming from the present node and move to the node corresponding to the closer decision vector, and so on until we find a terminal node, which will be $Q(\mathbf{x})$. The computational complexity of a binary TSVQ with depth L increases linearly with L and the storage increases exponentially with L. If a binary TSVQ with depth L is used as the fine vector quantizer, then the quantization is done in L comparisons as opposed to the M required by the brute force minimum distortion algorithm. There are also nonbinary TSVQ's.

Another possibility for the fine vector quantizer is a lattice code. The codebook for such consists of the points in some finite subset of a lattice in R^k. As such it is specified by the shape, size, and orientation of the lattice cells and by the outer boundaries of the subset. The simplest possible lattice code namely a cubic lattice code is a set $\{(x_1, \cdots, x_k) : x_i = n_i \delta, \text{ where } -N' \leq n_i \leq N', \text{ for all } i = 1, \cdots, k\}$, where δ is the edge length and N' is some positive integer. The minimum distortion quantization rule for such a lattice code is given by $Q(\mathbf{x}) = \mathbf{y}$ where $y_i = \psi(\frac{x_i}{\delta})\delta$,

$$\psi(x) = \begin{cases} -N' & \text{if } x < -N' + \frac{1}{2} \\ N' & \text{if } N' - \frac{1}{2} \leq x \\ N & \text{if } N - \frac{1}{2} \leq x < N + \frac{1}{2} \text{ for some integer } N \text{ such that } |N| \leq N' \end{cases}$$

Clearly, this algorithm works term by term on coordinates of a source vector $\mathbf{x}$. Therefore the computational complexity of the quantization algorithm is on the order of one operation per sample, independent of blocklength and number of points in the lattice code. It is also possible to consider more sophisticated lattices with better characteristics[4].

Experimental Results

We considered three sources: *iid* Gaussian, first-order Gauss-Markov, and samples of speech. All sources had mean zero and variance one. The Gauss-Markov source had normalized correlation coefficient of $\rho=0.8$.

For each source we sought the best possible codebook C for blocklengths $k=3,4,6,8$ for Gaussian sources, and $k=2,3,4$ for speech. The number of quantization vectors M was always chosen to be 2^k which corresponds to $R=1$ bit per sample. We used the generalized Lloyd algorithm with splitting based on training sequences from the given source[5]. For the Gaussian cases the training sequences were pseudorandomly generated. We were physically limited to sequences with at most 1,000,000 samples and used the rule of thumb 1,000 training vectors per quantization vector (hence, $1000kM$ samples). In the case of speech we had approximately 71,000 samples available for use. To generate sufficient training vectors we obtained approximately the same number of vectors by taking overlapping blocks.

For each source we designed a binary TSVQ from a source training sequence via the algorithm of [3]. Where feasible we made the tree as deep as M, the number of quantization vectors in C. If one were to go beyond depth M, then the TSVQ based fast quantization would require more computations than the brute force minimum distortion quantization algorithm. In addition, it was not computationally feasible to construct trees to depths greater than 14. The 2^L decision vectors at depths $L=\log_2 M, \log_2 M+1, \cdots, L_{\max}$ were each used as a fine codebook. For a given tree, as L increases the fine quantizer gets finer, so one expects $D(Q)$ to decrease at the cost of increased complexity. Recall that the computational complexity increases linearly with L and the required storage increases exponentially with L. The training sequences were again limited to at most 1,000,000 samples, and where possible we used 1,000 training vectors per decision vector at each depth.

At each step of the binary TSVQ quantization algorithm we need to compare a source vector with the two appropriate decision vectors. The fastest way of determining the closer decision vector is to form the dot product of the source vector with the vector pointing from one decision vector to the other and comparing this with a threshold. This implies a fifty percent reduction in both computational complexity and storage, when compared to calculating the distortion between the source vector and each of the two decision vectors and comparing the two distortions.

For each source and the codebook C described earlier we also tested our fast quantization approach based on a cubic lattice code. Specifically, for a given source, the codebook C designed for that source, and a size M' for C', we optimized the size of the basic cell or equivalently the outer boundaries of the lattice code via an iterative algorithm. It turned out that optimizing roughly amounted to choosing the cubic outer boundary to just barely cover all the codewords in C, regardless of the source distribution.

Distortions for various quantization rules (e.g. $D^*(C)$ and $D(Q)$) were computed by simulation on test sequences from the various sources. For Gaussian sources these pseudorandom test sequences were 1,000,000 samples long and where feasible were distinct from the training sequences. For the speech source, we used the 71,000 vectors mentioned above.

Table 1 shows the computational complexity and the amount of storage required by various quantization algorithms for a given k-dimensional codebook C with M codewords. Figures 1-5 summarize the results for the three sources. In each figure we have plotted the average distortion and the storage required by the three approaches: TSVQ based fast quantization (TBFQ), lattice based fast quantization (LBFQ), and brute force minimum distortion quantization (MDQ or FSVQ for full search vector quantization). The number next to each data point, if any, is the computational complexity of the corresponding quantization algorithm. There are no such numbers for the data points corresponding to LBFQ as the complexity is always one. In addition, we have included points corresponding to TSVQ's with rate 1 bit per sample used in their originally intended fashion for each source and each blocklength.

Discussion of Results

The following conclusions can be drawn from the figures:

1. In the case of shorter blocklengths ($k=2,3,4$) as C' gets finer (and the resulting storage of the fast quantization scheme increases) the distortion achieved by both TBFQ and LBFQ approach that of the FSVQ at a reasonable rate, and LBFQ seems to be better than TBFQ in an over all sense. For example, for speech with $k=4$ and LBFQ we get a factor of 13 increase in storage over FSVQ, a 16 fold decrease in computations, and 10% increase in distortion.
2. For larger blocklengths ($k=6,8$) the distortions achieved by TBFQ and LBFQ do not converge as rapidly to that of FSVQ. One possible reason is the increased dimensionality of the space may require a higher ratio of M' to M. TBFQ becomes a more viable option than LBFQ for sources with memory, because the TSVQ used in TBFQ is specifically designed for the given source and so its M' quantization vectors are better placed in R^k (they match the source *pdf*) than those of a lattice code.
3. If TSVQ has distortion close to that of FSVQ, it's hard to beat, for example in the cases of Gaussian *iid* and Markov sources. It outperforms TBFQ in such cases mainly because it has smaller computational complexity and storage requirement. However, LBFQ is still competitive. At roughly the same distortion as TSVQ, it offers less complexity (its complexity is one) at the cost of some increase in storage.
4. When there is a significant gap between the distortions of FSVQ and TSVQ, both TBFQ and LBFQ become more attractive, for example, for speech with $k=2,3,4$. Although it has the best distortion, FSVQ would become computationally infeasible for large k. On the other hand, it can be seen from Figure 5 that there are codes with less distortion and complexity than TSVQ and comparable storage. There are TBFQ and LBFQ schemes that offer a further decrease in distortion at the cost of some increase in storage and in the case of TBFQ at the cost of some increase in computational complexity as well.

References

1. R. M. Gray, "Vector Quantization," IEEE ASSP Magazine, vol. 1, April 1984, pp. 4-29.
2. P. C. Chang, J. May, and R. M. Gray, "Hierarchical Vector Quantizers with Table-Lookup Encoders," 1985 IEEE International Communications Conference, vol. 3, pp. 1452-1455.
3. A. Buzo, A. H. Gray, Jr., R. M. Gray, and J. D. Markel, "Speech Coding Based upon Vector Quantization," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-28, Oct. 1980, pp. 562-574.
4. J. H. Conway and N. J. A. Sloane, "Voronoi Regions of Lattices, Second Moments of Polytopes, and Quantization," IEEE Transactions on Information Theory, vol. IT-28, March 1982, pp. 211-226.
5. Y. Linde, A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design," IEEE Transactions on Communications, vol. COM-28, Jan. 1980, pp. 84-95.

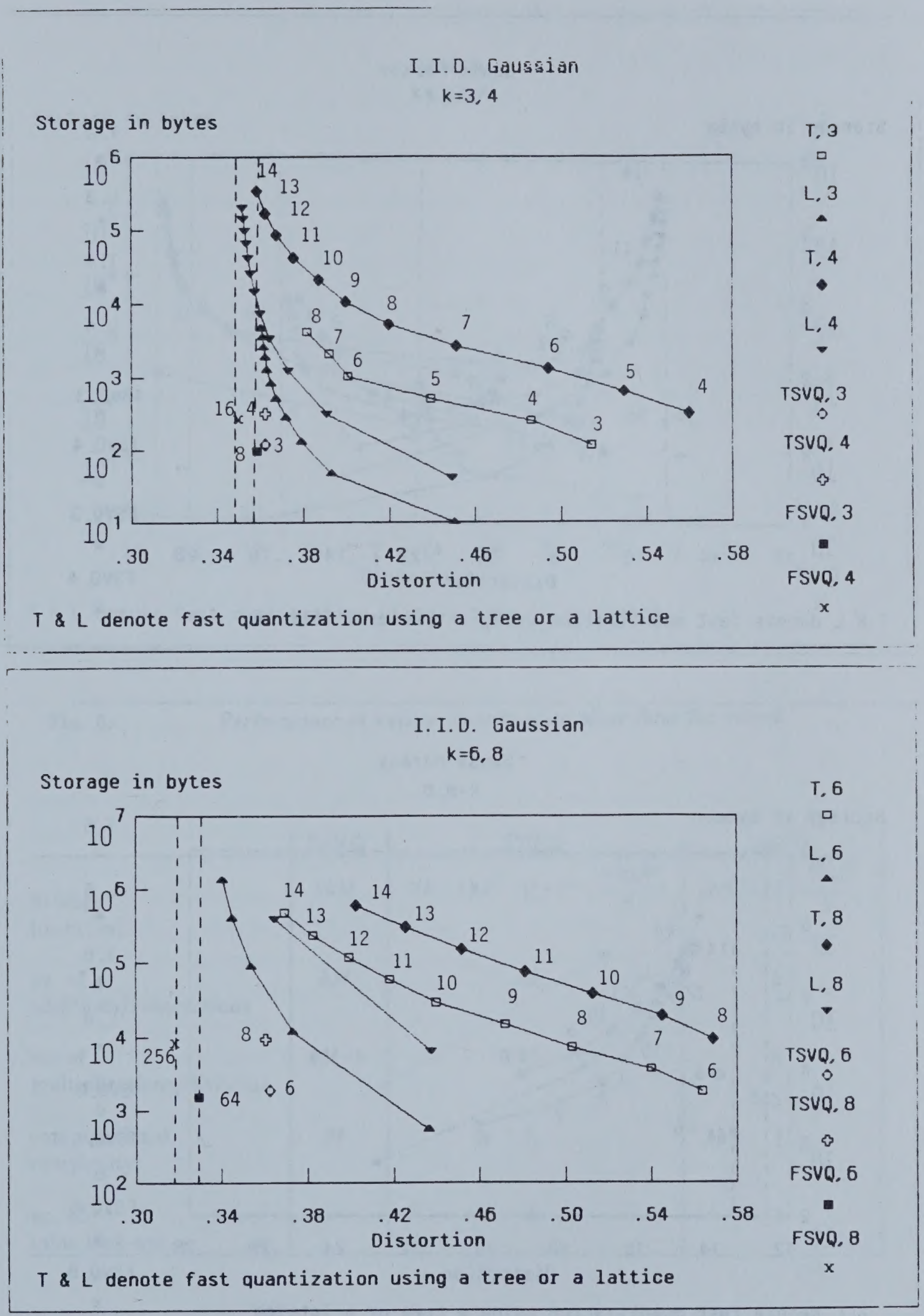


Fig. 1-2: Performance of various quantization algorithms for *iid* Gaussian source.

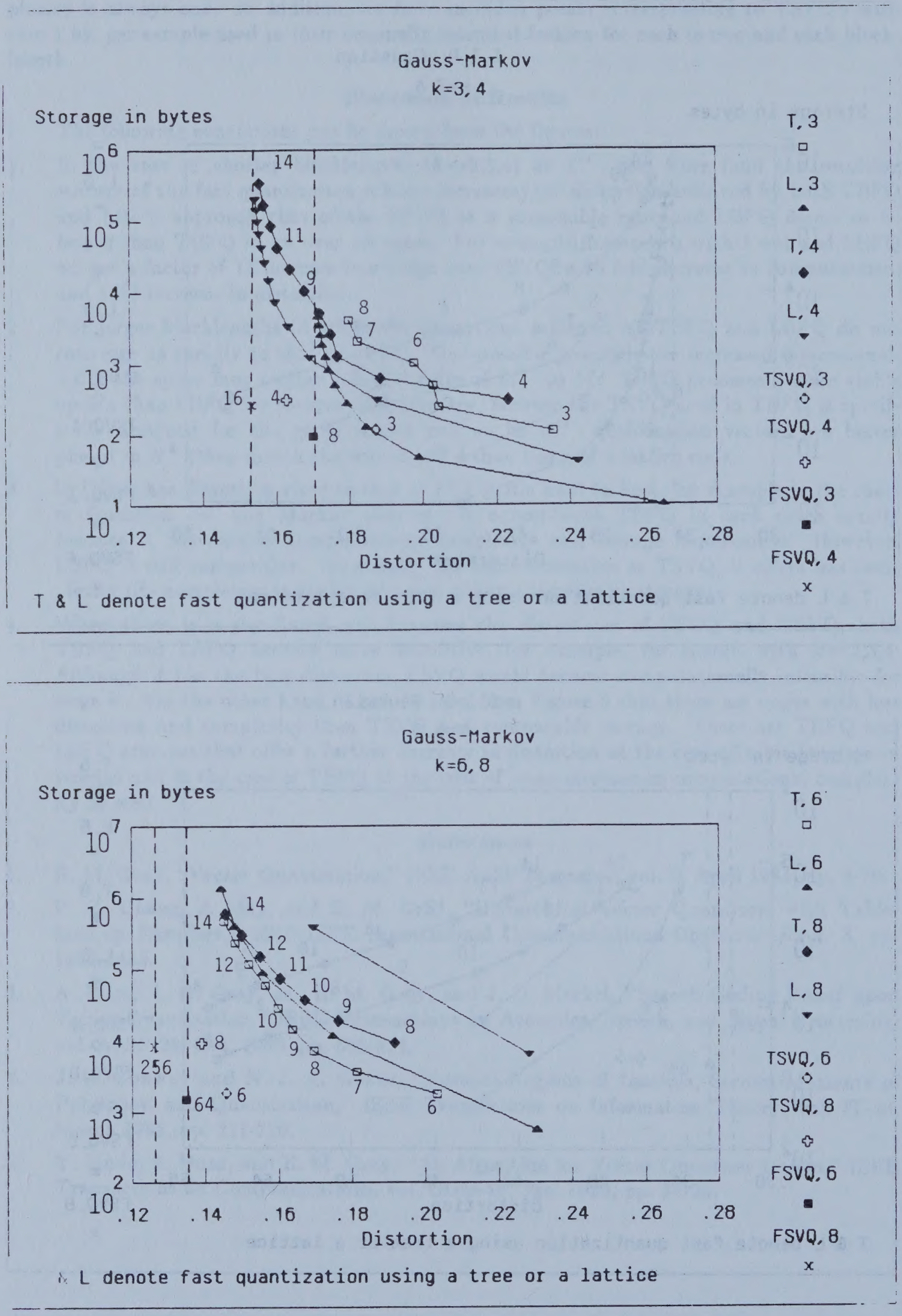


Fig. 3-4: Performance of various quantization algorithms for Gauss-Markov source.

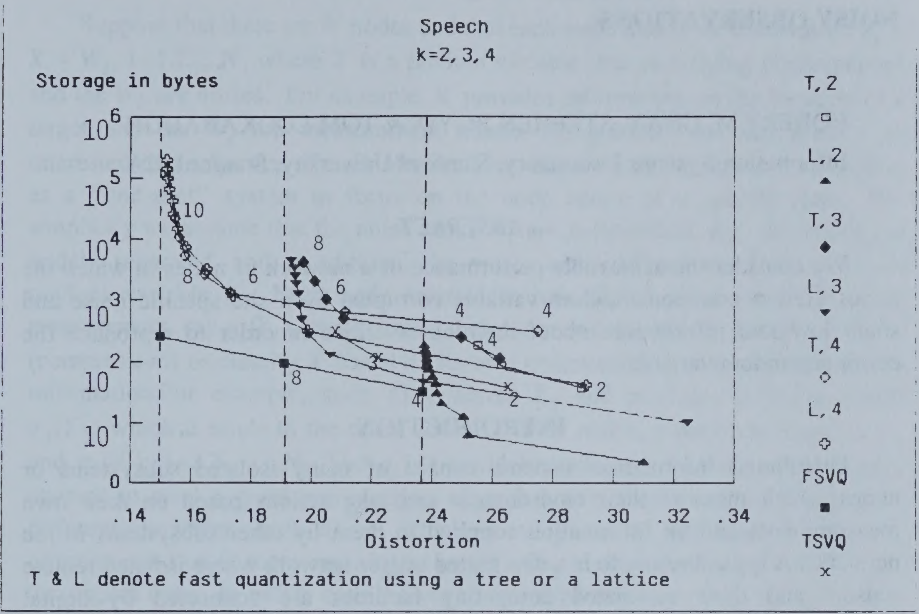


Fig. 5: Performance of various quantization algorithms for speech.

	FSVQ	TBFQ	LBFQ
Storage (in bytes)	$4kM$	$4(k+1)(2^L-1)+2^L\cdot\frac{\log_2 M}{8}$	$(2N'+1)^k\cdot\frac{\log_2^M}{8}$
no. of additions/subtractions	kM	kL	k
no. of multiplications/divisions	$kM-1$	kL	k
computational complexity	M	L	1
no. of table look-ups	—	1	1

Table 1. Storage and Computational Complexity for various Quantization algorithms.

LOW RATE DISTRIBUTED QUANTIZATION OF NOISY OBSERVATIONS

ROBERT M. GRAY, STEPHEN BOYD, & TOM LOOKABAUGH
Information Systems Laboratory, Stanford University, Stanford, CA

ABSTRACT

We consider the achievable performance of a network of nodes in which the nodes view a common random variable corrupted by a site specific noise and share low rate information about their observations in order to reproduce the common random variable.

INTRODUCTION

Distributed information systems consist of many isolated subsystems or nodes which measure their environment and take actions based on their own measurements and on information supplied to them by other subsystems in the network. A typical example is a distributed sensor network where isolated remote sensors and their associated computing facilities are connected by digital communication links; the separate sensors must use their own observations together with shared digital side information from other sensors in order to perform reliable estimation or detection on objects moving through the network.

Distributed compression and classification systems form a special case. An example is the following: Many separate nodes make measurements of a common phenomenon with different measurement noise and share digital information based on these measurements so that each node can combine its observation with the data from the other nodes in order to well reproduce (compression) or make a decision about (classification) the common phenomenon. Such a system can be viewed as a source coding analog of the broadcast network considered by Gallager [5]. Results providing Shannon theory bounds and simple code design techniques have been developed for the special case of only two nodes [1,2,3,4]. It is of interest, however, to consider the case where there are a large number of nodes, but the permitted communication rate is small, say one bit. The goal is to find the achievable performance for a fixed block size asymptotically as the number of nodes grows. This is in distinction to the usual asymptotic quantizer theory which considers asymptotically large rate for fixed block size or the Shannon theory which considers asymptotically large dimension for fixed rate. In this paper we consider a simple example of such a system.

DISTRIBUTED QUANTIZATION

Suppose that there are N nodes and that each node makes an observation $Y_i = X + W_i$, $i=1,2,...,N$, where X is a random variable (the underlying phenomenon) and the W_i are noises. For example, X provides information on the location of a target and the W_i are measurement noises. In practice we will usually be interested in a sequence of such observations, but we here formulate the problem as a "one-shot" system to focus on the node action at a specific time. For simplicity we assume that the noise terms W_i are independent, e.g., the nodes are widely separated, and in addition, the noises are independent of the common random variable X. Each node sends low rate digital information about its measurement, say $q_i(Y_i)$, to all of the other nodes and attempts to either reproduce (compression) or classify X based on its own observation and the shared low rate information.For example, node #1 observes Y_1 and produces a digital signal $q_1(Y_1)$ which it sends to the other nodes. It then makes a decision based on Y_1 and $q_i(Y_i)$; $i=2,3,\cdots,N$. The $q_i(y)$ could be ordinary quantizers and simply attempt to send a good reproduction of y, but they might do something quite different, however, such as to extract some piece of information from the observations that is more meaningful to the other nodes when pooled with all of the digital data. In either case, however, we shall refer to the q_i as quantizers since they map analog variables into binary variables.

As a related problem, each node might only be able to use its own quantized output and hence must base its decision on $q_i(Y_i)$; $i=1,\cdots,N$. In this case all of the nodes use the same information to reach their final decision. Clearly this system provides a bound on the performance of the previous system since the decision makers have less information. If one has a distortion measure $d(x,\hat{x})$ which measures the distortion of reproducing x as $\hat{x}$, the common design goal of information theory is to minimize for each node the average distortion $E(d(X,\hat{X}(Y_i,q_j(Y_j);j\neq i)))$, where $\hat{X}$ is the reproduction of the original X based on the available information. By appropriate choice of the distortion measure, one can also treat the problem of classification of which of a finite collection of possible distributions produced X (this is essentially what vector quantization of linear predictive coded speech does).

Suppose that the quantizers are all constrained to have only 1 bit of information, that is, they are binary quantizers. One strategy would be to make all of the quantizers do as good a job at reproducing their observation with minimum average distortion (using, e.g., the Lloyd-Max algorithm), but this may not be good given the overall goal of sharing information to reproduce the underlying variable. A natural question is the following: Does there exist a strategy such that the average distortion between the guess based upon the shared information and the true value tends to 0 as $N \rightarrow \infty$? On one hand, one is getting an infinite number of bits in the limit and one would expect that that would yield ever smaller distortions, eventually converging to zero. On the other hand, each bit provides information about a random vector corrupted by continuous noise and hence in a sense there is more noise than 1 bit per node can handle. We shall argue below that at least in a special case, as $N \to \infty$ an appropriate choice of quantizers yields an arbitrarily small average distortion at each node, even though the rate of each node is only 1 bit per vector.

AN EXAMPLE

For simplicity we consider the scalar case so that the observation is a real random variable X. We also make the following simplifying assumptions:

1. The noises W_i are independent, identically distributed, and nonnegative, and EW is known (in particular, W has an expectation).
2. The random variable X is nonnegative, independent of the W_i, and $EX < \infty$.
3. The distortion measure is the squared error $d(x,\hat{x}) = (x-\hat{x})^2$.

Theorem: If 1. through 3. above hold, then there is a sequence of 1 bit quantizers $q_i^{(N)}$, $i=1,\cdots,N^2$ and decoding rules $\hat{X}_N = f(q_i^{(N)}(Y_i),\ i=1,\cdots,N^2)$, such that

$$\lim_{N \to \infty} Ed(X,\hat{X}_N) = 0 ,$$

that is, by choosing the number of nodes N^2 large enough the average distortion for each node can be made arbitrarily small.

Sketch of Proof: For each N define for the ith node the quantizer

$$q_i^{(N)}(y) = 1_{(-\infty, i/N]}(y) ,\ i=1,\cdots,N^2 ,$$

where $1_F(y)$ is the indicator function for F, that is, is 1 if $y \in F$ and 0 otherwise. Define the decoder function

$$\hat{X}_N = \frac{1}{N}\sum_{i=1}^{N^2} (1-q_i^{(N)}(y_i)) - EW . \tag{1}$$

For the moment consider conditional expectations with $X=x$ fixed. We have

$$E(\hat{X}_n \mid X=x) = \frac{1}{N}\sum_{i=1}^{N^2} Pr(W+x > i/N) - EW . \tag{2}$$

Using the fact that $Pr(W+x > z)$ is monotone in z, it can be shown that this sum converges as $N \to \infty$ to the integral

$$E(\hat{X}_N \mid X=x) \underset{N \to \infty}{\to} \int_0^\infty (1-F_{W+x}(w))dw - EW = x ,$$

where F_{W+x} is the cumulative distribution function of $W+x$. Thus the estimate $\hat{X}_N$ is an asymptotically unbiased estimate of x.

Considering next the conditional variance, we have that

$$\sigma^2_{\hat{X}_N^2 | X=x} = \frac{1}{N^2} \sum_{i=1}^{N^2} Pr(x+W > i/N)(1-Pr(x+W > i/N))$$

$$\leq \frac{1}{N^2} \sum_{i=1}^{N^2} Pr(x+W > i/N) \underset{N\to\infty}{\rightarrow} 0 .$$

since the sum in (2) converges to $EW + x$.

Considering conditional L_2 norms we have using the triangle inequality that

$$||\hat{X}_N - x|| \leq ||\hat{X}_N - E(\hat{X}_N | X=x)|| + | E(\hat{X}_N | X=x) - x | \underset{N\to\infty}{\rightarrow} 0$$

which proves that

$$\lim_{N\to\infty} E(d(X, \hat{X}_N) | X=x) = 0 \tag{3}$$

for all x. It can be shown that $E(\hat{X}_N | X=x) \leq x$ for all x, so by dominated convergence we conclude from (3)

$$\lim_{N\to\infty} Ed(X, \hat{X}_N) = \lim_{N\to\infty} \int E(d(X, \hat{X}_N) | X=x) dPr(X < x) = 0$$

and the theorem is proved.

COMMENTS

The above simple example is new and makes several interesting points. The first point is that an intelligent quantizer strategy involves nodes cooperating to extract certain information from the observations and not acting independently to simply quantize their observations. The second point is that such a strategy can yield arbitrarily small average distortion in the limit of many nodes. The third point is that several of the nodes might fail to provide their bits without affecting the asymptotic properties since the nodes make overlapping tests. The estimate is simply modified to not include the missing information. Provided there are enough nodes, the overall approximation is still good. This robustness against missing nodes can be extremely valuable in a distributed system where nodes may fail.

There are many generalizations of possible interest: The nonnegative restriction can be removed in a straightforward manner. The generalization to noise values that are not independent is more important and less obvious. Intuitively one would expect that reasonable models would entail highly correlated noise in adjacent nodes with the correlation going to 0 with increased distance. It is also desirable to remove the assumption of identical distributions, although they must be related in order for the cooperative information to help. Generalizations to vector quantization in the presense of independent vector noises have been obtained using the ergodic theorem and quantizers that are indicator functions for sets that partition the space in an appropriate manner.

REFERENCES

1. T.J. Flynn, "Quantizer design for distributed sensing," Ph.D. Thesis, Stanford University, December 1984.
2. T.J. Flynn and R.M. Gray, "Quantizer Design for Correlated Observations", **Abstracts of Papers**, 1985 IEEE International Symposium on Information theory, Brighton, England, June 1985, p. 117.
3. T.J. Flynn and R.M. Gray, "Achievable Rates and Distortion for Correlated Observations", Ibid., p. 153.
4. T.J. Flynn and R.M. Gray, "Encoding of correlated observations," submitted for possible publication. (1985)
5. R.G. Gallager, "Finding parity in a broadcast network," **Abstracts of Papers**, 1985 IEEE International Symposium on Information Theory, June, 1985, p. 116.

ACKNOWLEDGEMENTS

This research was supported by the ARPA Distributed Information Theory Contract N00039-84-C-0211 Task 6.

Performance of Entropy-Constrained Block Transform Quantizers[1]

by

N. Farvardin and F. Y. Lin
Electrical Engineering Department
and
Systems Research Center
University of Maryland
College Park, Maryland 20742

Summary

There are numerous techniques available for analog-to-digital conversion, or data compression, of discrete-time processes. They range from simple scalar (zero-memory) quantization or PCM, to more sophisticated schemes such as predictive encoding, tree encoding and multi-dimensional quantization. It is well-known that the potential gains of data compression is most noticeable for highly correlated sources.

One of the most popular data compression schemes for encoding of correlated sources is block transform quantization. The main advantages associated with the block transform quantization scheme are: (i) good performance results, and (ii) ease of implementation.

In this paper we study the rate-distortion theoretic performance of optimal block transform quantization schemes on first-order stationary autoregressive processes. More precisely, we assume that blocks of length L of source outputs are operated upon by the Karhunen-Loeve transformation. The transform coefficient are then quantized and coded by means of *optimal entropy-constrained* zero-memory quantizers operating at rates $H_1, H_2, \ldots, H_L$. The problem that we address is that of determining these rates in such a way that the overall average squared-error distortion is minimized.

This problem, which is very much similar to that studied by Huang and Schultheiss [1], can be formulated as the following constrained nonlinear programming problem:
Minimize

$$D(\underline{H}) = \frac{1}{L}\sum_{i=1}^{L} \lambda_i D_i(H_i) , \tag{1.a}$$

subject to

$$\frac{1}{L}\sum_{i=1}^{L} H_i \leq R , \tag{1.b}$$

in which $\underline{H} = [H_1, H_2, \ldots, H_L]$, λ_i is the variance of the ith coefficient and $D_i(H)$ is the variance-normalized distortion-rate performance of the optimum entropy-constrained

---

[1] This work was supported in part by a grant from Martin Marietta Laboratories, in part by a grant from the Minta Martin Fund for Aeronautical Research and in part by the General Research Board of the University of Maryland.

quantizer [3] operating on the *ith* transform coefficient. The distortion criterion is the squared-error measure throughout. The main distinction between this work and the previously reported results on block transform coding is in the quantization operation. In previous works [1],[2], Lloyd-Max quantization was employed for the encoding of the transform coefficients and hence the problem was that of optimal bit assignment among various transform coefficients.

In this work we confine ourselves to uniform-threshold quantization [3]. With this constraint, we have used Kuhn-Tucker conditions to solve the nonlinear constrained optimization problem described in (1). We have developed a simple algorithm for determining the best rate assignment and quantizer design. This algorithm is applied to first-order Gauss-Markov and Laplace-Markov sources [4]. Our results indicate that in all cases noticeable performance improvements will be obtained over the scheme described in [1]. These improvements become more noticeable at higher bit rates. Also, performance improvements are larger for the Laplace-Markov source. Specifically, for $L=4$ and $R=3$ bits/sample, an SNR improvement of 4.4 dB is obtained for a Laplace-Markov source with correlation coefficient 0.9.

Furthermore, we have compared our results against the DPCM performance results [4] and the vector quantization results [5]. Our results in all cases supercede those of vector quantization [5]. This is interesting as vector quantization is far more complex to design and implement. Also, for the Gaussian source, our scheme offers performance improvements over DPCM for low bit rates. This is not the case in the Laplacian case. Detail of this work including extensive numerical results can be found in [6].

References:

1. J. J. Y. Huang and P. M. Schultheiss, " Block Quantization of Correlated Gaussian Random Variables", *IEEE Trans. Commun. Syst.*, Vol. CS-11, pp. 289-296, Sept. 1963.
2. N. S. Jayant and P. Noll, *Digital Coding of Waveforms: Principles and Applications to Speech and Video*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1984.
3. N. Farvardin and J. W. Modestino, " Optimum Quantizer Performance for a Class of Non-Gaussian Memoryless Sources", *IEEE Trans. Inform. Theory*, Vol. IT-30, pp. 485-497, May 1984.
4. N. Farvardin and J. W. Modestino, " Rate-Distortion Performance of DPCM Schemes for Autoregressive Sources", *IEEE Trans. Inform. Theory*, Vol. IT-31, pp. 402-418, May 1985.
5. R. M. Gray and Y. Linde," Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources", *IEEE Trans. Commun.*, Vol. COM-30, pp. 381-389, Feb. 1982.
6. N. Farvardin and F. Y. Lin," Performance of Entropy-Constrained Block Transform Quantizers", to be submitted to *IEEE Trans. Inform. Theory*, for publication.

SOME FAMILIES OF ZERO-ERROR CODES FOR THE 2-USER BINARY ADDER CHANNEL WITH FEEDBACK

Zhen Zhang and Toby Berger
Center for Applied Mathematics and
School of Electrical Engineering
Cornell University,
Ithaca, NY 14853

James L. Massey
Institute of Telecommunications
Swiss Federal Institute of Technology
ETH-Zentrum
Zurich, Switzerland

ABSTRACT

We introduce families of zero-error codes for the real binary adder channel with feedback that achieve high rate pairs. First we produce two families of zero-error block codes for the case in which only one of the two senders receives feedback about the channel output. In the first of these families the uninformed sender transmits at a rate of nearly 1 bit per symbol and the informed sender transmits slightly less that 1/2 bit per symbol. The second family is designed for the case in which the informed sender sends at or near one bit per symbol and the uninformed one sends nearly 1/2 bit per symbol. We also introduce a family of zero-error codes based on the Fibonacci recursion that are readily implemented by means of a simple square-dividing strategy. The Fibonacci codes achieve $R_1 = R_2 = \log_2(\frac{1+\sqrt{5}}{2})$ in the limit of large block length. Time-sharing between members of these three code families yields an achievable rate region, or inner bound, to the zero-error capacity region for block coding.

For the case in which the feedback is available to both senders, we use a variant of the Fibonacci difference equation to generate zero-error block codes with slightly higher asymptotically equal rate codes.

1. INTRODUCTION

Consider the two-access communication system shown in Figure 1. Two independent sources wish to send information to the receiver. During a message interval, the messages emanating from the sources are encoded independently with two binary block codes of the same length n. We assume that we have bit and block synchronization.

The two binary input vectors x and y are transformed by the channel into an output vector $z = x + y$, where the plus sign denotes addition over the reals. This so-called "binary adder channel" is a special case of the multiple access channel. Block coding for this channel has been studied by several authors [1]-[5]. Its Shannon capacity region has been determined, and many results have been obtained about uniquely decodable (i.e., zero-error) codes for it. We shall design some uniquely decodable block codes for cases in which z is fed back either to one or to both senders.

If both encoding functions depend on the previus channel outputs, we say we have full feedback. If only one of them does, we say we have partial feedback. We concentrate mainly on the partial feedback case. One reason for this is that, in the full feedback case, variable length codes perform significantly better than block codes (cf. [1]), assuming one is willing to tolerate occasional long coding delays.

Let us now describe the encoding procedure for the two-access channel with partial feedback shown in Figure 1. The informed encoder's encoding function f_k for the k^{th} time slot depends both on the message m_2 it is trying to send and on the channel outputs during the first $k-1$ time slots. That is, $y_k = f_k(m_2, z_1, \cdots, z_{k-1})$. The uninformed encoder's output x_k during the k^{th} time slot depends only on the message m_1. In the full feedback case we have $x_k = g_k(m_1, z_1, \cdots, z_{k-1})$.

This work was suported in part by NSF Grant ECS-8305681.

2. TWO FAMILIES OF CODES FOR BINARY ADDER CHANNEL WITH PARTIAL FEEDBACK

(a) The sets $W(n, k)$

For any $x \in \{0,1\}^n$, let

$$t(x) = |\{i : x_i \neq x_{i-1}, i > 1\}|$$

denote the number of transitions in x. Define

$$W(n, k) = \{x \in \{0,1\}^n : t(x) = k\}.$$

(b) The i^{th} run $l_i(x)$.

The segment $(x_j, \cdots, x_{j+s})$ of x is denoted by $l_i(x)$ and called the i^{th} run if

1) $x_j = \cdots = x_{j+s}$,

2) $x_{j-1} = x_{j+s+1} \neq x_j$,

3) $|\{s < j : x_s \neq x_{s+1}\}| = i-1$.

(c) The first family of codes

The strategy of the first encoder, who receives no feedback, will be simply to transform his message m_1 into a word $x \in W(n, k)$. The second encoder, who is privy to the feedback, maps his message m_2 into a word $y \in W(s, t)$ which he then sends as follows. Let

$$y = (y_1, \cdots, y_s).$$

Define $f(0) = 0, f(1) = 2$. The second encoder keeps sending y_1 until it receives a feedback $f(y_1)$. Then it keeps sending y_2 until it receives a feedback $f(y_2)$, and so on. If and when it finishes with y, the second encoder keeps sending y_s.

The decoder receives $z = (z_1, \cdots, z_n) \in \{0,1,2\}^n$. Without loss of generality denote the positions of the non-1 components of z by $a_1, a_2, a_3, \cdots$. If the number of the entries of this sequence is s or bigger, then define the second decoding function by

$$\hat{y}(z) = (f^{-1}(z_{a_1}), \cdots, f^{-1}(z_{a_s})),$$

and the first decoding function by

$$\hat{x}(z) = (\hat{x}_1, \cdots, \hat{x}_n),$$

where

$$\hat{x}_j = f^{-1}(z_{a_{l-1}})$$

for $a_{l-1} \leqslant j < a_l$.

It is easy to check that a sufficient condition for this code to be uniquely decodable is that, for any $x \in W(n, k)$ and $y \in W(s, t)$, the length of the a-sequence is at least s. This is because in this case, and only in this case, can the second encoder finish sending y within n slots. To ensure that the a-sequence has length at least s, we need one transition in x for each transition in y and at most two transitions in x for each non-transition in y. Thus, the condition that guarantees unique decodability is

$$2(s - t) + t = 2s - t \leqslant k . \tag{*}$$

(d) Rate pairs.

The rate for the first encoder is

$$\frac{1}{n} \log_2 \binom{n-1}{k}$$

and the rate for the second encoder is

$$\frac{1}{n}\log_2\binom{s-1}{t}.$$

For large n, s and t, let $k/n = p$, $t/s = q$, and $s/n = r$. Then we have

$$R_1 = h(p) \text{ and } R_2 = rh(q),$$

and (*) reads

$$2r - rq \leqslant p. \tag{**}$$

Using equality in (**), we have

$$r = \frac{p}{2-q}.$$

Therefore, the second rate is

$$\frac{p}{2-q}h(q).$$

If $p = 0.5$, then $R_1 = 1$, in which case the highest rate for the second encoder under the constraint (**) is $R_2 = 0.347$. The highest rate sum $R_1 + R_2$ reached by this family of codes is $2\log_2\frac{\sqrt{5}+1}{2} = 1.3885$. The rates of this code family are depicted in Fig 2.

(e) The second family of codes

For given n and k, we construct a code from the set $W(n,k)$. For any $x \in W(n,k)$, let $|l_i(x)|$ be b_i. Define

$$v(x) = (\overbrace{1,\cdots,1}^{b_1},0,0,\overbrace{1,\cdots,1}^{b_2},0,0,1,\cdots)$$

This is a binary sequence in $W(n+2k, 2k)$ which consists of k runs of 1's whose lengths are the b_i's separated from one another by pairs of consecutive 0's. The first encoder sends the sequence $v(x)$ for some $x \in W(n,k)$. The second encoder continually uses the feedback to recover the sequence the first encoder already has sent. The second encoder sends a sequence in $\{0,1\}^{n+k}$ into which it injects a 0 whenever the feedback indicates that the first encoder has begun one of its pairs of consecutive 0's in the previous slot. The decoder is able to recover the sequences sent by the both encoders because it receives 0's only either in isolation or in runs of length 2. It knows that each of the zero pairs sent by the first encoder ends either at a received isolated 0 or at the end of a received pair of 0's. Thus, the decoder is able to recover the sequence sent by the first encoder. Using that sequence, the decoder then can recover the sequence transmitted by the second encoder and expunge from it the k extra 0's that the second encoder injected.

The rates of this code are

$$R_1 = \frac{1}{n+2k}\log_2\binom{n}{k}$$

and

$$R_2 = \frac{n+k}{n+2k}.$$

Letting $k/n = p$, we have for large n that

$$R_1 = h(p)/(1+2p)$$

and

$$R_2 = (1+p)/(1+2p).$$

This family of achievable rate pairs also is sketched in Figure 2. Numerical result shows that the best rate sum of any code in this family is 1.375.

3. CODES GENERATED BY DIFFERENCE EQUATIONS

(a) Square dividing strategy

Encoding for the binary adder channel with full or partial feedback also can be described by means of square-dividing strategies. Assume that the message sets at the two encoders are $C_t, t = 1, 2$, where $C_1 = \{1, \cdots, a\}$ and $C_2 = \{1, \cdots, b\}$. We consider the set $C_1 \times C_2$. In the first slot for a message in a certain subset of C_1, say $C_1^1(0)$, send 0 ; for a message in the set $C_1^1(1) = C_1 - C_1^1(0)$, send 1. Similarly, for the second encoder, define $C_2^1(0)$ and $C_2^1(1)$, and send 0 and 1, respectively. Thus, the square $C_1 \times C_2$ is divided into four subsquares with outputs 0,1, and 2 as shown in Figure 3.

Temporarily confine attention to the full feedback case. In this case after receiving the first feedback each encoder can identify which of these four subsquares was employed during the first slot. They then divide it into smaller subsquares during the second slot, and into finer and finer subsquares during subsequent slots. By observing the sequence of channel outputs, the decoder knows the input is a message pair in the group of subsquares with these outputs. The decoder eventually can make the correct decision once there is only one message pair that is consistent with the channel output sequence. We shall continue to consider only cases in which no error is allowed. Kasami and Lin [3] call such zero-error codes "uniquely decodable". In the square-dividing terminology unique decodability means that eventually the square is divided into $a \cdot b$ subsquares each of which has a unique channel output sequence.

Figure 4 is an example of the uniquely decodable code with sizes $a = 5$ and $b = 3$. Note that in this example the first encoder always sends either 111 for message 1, 110 for message 2, 100 for message 3, 001 for message 4 or 000 for message 5. Hence, the first encoder need not be privy to the feedback. In slot 1 the second encoder sends 1 for either message 1 or message 2 and sends 0 for message 3. In the second slot for either message 1 or message 3 a 1 is sent if the larger of the two possible feedback symbols was received and a 0 is sent if the smaller one was received; the opposite is done for message 2. In the third and final slot a 1 is sent unless the feedback pair from the first two slots indicates that the first encoder is trying to send the third of its five possible messages, in which case a 0 is sent. A unique ternary output sequence results for each of the 15 possible message pairs. This is but one of many different zero-error coding strategies that could have been employed in this example. The rationale underlying this particular strategy is explained in Section (b) below in conjunction with the proof of Theorem 1'.

A pair of numbers (a, b) is called *k-attainable* (for the full feedback case or the partial feedback case) if there exists a uniquely decodable code (for the corresponding case) of length k with codeword sets of sizes a and b. Clearly, any pair that is k-attainable for the partial feedback case must be k-attainable for the full feedback case. Of course, if (a, b) is k-attainable, $c \leqslant a$, and $d \leqslant b$, then (c, d) also is k-attainable. As shown by the example, (5,3) is 3-attainable with partial feedback. For small k it is not difficult to determine whether or not a pair of numbers is k-attainable, but as k grows this task becomes imposing.

We introduce a method to generate families of attainable pairs. We call the codes we use to reach these pairs codes generated by difference equations. As their name implies, they have a recursive construction that makes them easy to encode and decode. Because of their high rates and ease of implementability, we feel they are an important family of codes.

(b) Fibonacci codes

The simplest of our codes generated by difference equations are the so-called Fibonacci codes. Let $\{a_i\}$ be the Fibonacci numbers defined by

$$a_i = a_{i-1} + a_{i-2} \tag{1}$$

with $a_0 = a_1 = 1$. We prove below that (a_i, a_{i+1}) is i-attainable.

A pair of difference equations

$$a_i = \phi(a_j, j < i), \tag{2a}$$

$$b_i = \psi(b_j, j < i). \tag{2b}$$

will be called a pair of *code generating equations* if there exist initial conditions and a positive integer S such that the sequences $\{a_i\}$ and $\{b_i\}$ generated by these equations and initial conditions have the property that (a_i, b_i) is $(S+i)$-attainable for all $i \geqslant 1$. Thus, we have claimed that the pair of Fibonacci equations $a_i = a_{i-1} + a_{i-2}$ and $b_i = b_{i-1} + b_{i-2}$ are code generating equations for $a_0 = a_1 = b_0 = 1, b_1 = 2$ and $S = 0$.

It is easy to find code generating equations, but at present we have no general way of finding ones that possess high rates. We shall show, however, that the aforementioned Fibonacci codes and another set of code generating equations we introduce in Section 4 do indeed achieve high rates.

We now prove our claim that a pair of Fibonacci equations are code generating for the full feedback case. Subsequently, we extend this result to the partial feedback case. To facilitate the proof, we introduce the concept of an *attainable cluster*. A union of disjoint rectangles that share the same output sequence is called a cluster. For example, in the (5,3) code of Figure 4, after the first step the 2x2 rectangle in the upper right hand corner and the 1x3 rectangle in the lower left hand corner constitute a cluster. A cluster is called *k-attainable* if after k or fewer further divisions, it can be reduced to single "points" each of which has a distinct output sequence. The cluster comprised of the aforementioned 2x2 and 1x3 rectangles is 2-attainable. It should be obvious that a cluster of two rectangles of sizes of 1x2 and 1x1 is 1-attainable.

Theorem 1. A pair of Fibonacci equations are code generating for full feedback.

Proof. First, we define two types of parameterized clusters and prove that, by one step of square dividing, each of them can be reduced to clusters of the same two types with smaller parameter values. The first cluster type is a union of two rectangles with sizes $a_k \times b_{k-1}$ and $a_{k-1} \times b_k$; the second is a rectangle with size $a_k \times b_k$. We denote them respectively, by

$$\lambda_k = a_k \times b_{k-1} \vee a_{k-1} \times b_k \tag{3}$$

and

$$\mu_k = a_k \times b_k , \tag{4}$$

where $a \times b$ denotes an a by b rectangle and V denotes the union of disjoint sets. Note that we can divide λ_k into three parts,

$$\lambda_k = [\mu_{k-1}]_2 \vee [\lambda_{k-1}]_1 \vee [\mu_{k-1}]_0 , \tag{5}$$

where $[\mu_{k-1}]_2$ means that the set with output 2 is μ_{k-1}, and so on.

For μ_k we have

$$\mu_k = [\mu_{k-1}]_2 \vee [\lambda_{k-1}]_1 \vee [\mu_{k-2}]_0 . \tag{6}$$

Since the 1-attainability of both $\lambda_1 = 1 \times 1 \vee 1 \times 2$ and $\mu_1 = 1 \times 2$ are obvious, the theorem is proved.

The limiting rates of the Fibonacci code family are

$$R_1 = R_2 = R_f = \lim_{k \to \infty} \frac{1}{k} \log_2 a_k = \log_2 \left(\frac{\sqrt{5}+1}{2}\right) = 0.694. \tag{7}$$

Now we show that the Fibonacci codes actually are implementable in the partial feedback case.

Theorem 1'. A pair of Fibonacci equations are code generating for partial feedback.

Proof. We need to prove that the Fibonacci encoding strategy can be implemented with one of the two encoders not having access to the feedback. That is, we must exhibit a technique by means of which the uninformed encoder can correctly divide each of the clusters that appears in the square dividing procedure into 1-subsets and 0-subsets. Note, as shown in Figure 5, that the sizes of the horizontal edges of the subsquares after the successive square-divisions are

Originally (Figure 5a): a_k

After one division (Figure 5b): a_{k-1} , a_{k-2}

After two divisions (Figure 5c): a_{k-2} , a_{k-3} , a_{k-2}

After three divisions (Figure 5d): $a_{k-3}, a_{k-4}, a_{k-3}, a_{k-3}, a_{k-4}$

After four divisions : $a_{k-4}, a_{k-5}, a_{k-4}, a_{k-4}, a_{k-5}, a_{k-4}, a_{k-5}, a_{k-4}$

and so on. Observe that at the i^{th} step each of the sizes in question is either a_{k-i} or a_{k-i-1}.

Define $u_i = 1$ if the message to be sent by the encoder without feedback is a member of a subset of size a_{k-i}; otherwise, define $u_i = 0$. The strategy of the encoder without feedback is to send

$$x_i = u_i u_{i-1}$$

at the i^{th} step. This strategy is depicted in Figures 6a and 6b, the former of which shows the subset sizes arranged on successive levels of a tree and the latter of which shows the corresponding binary transmissions.

Analogously define the binary function v_i of the sizes of the vertical squares at the i^{th} step by $v_i = 1$ for subsets of size b_{k-i} and $v_i = 0$ for subsets of size a_{k-i-1}. The strategy of the encoder with feedback is to send

$$y_i = v_i \oplus 1 \oplus u_{i-1}$$

at the i^{th} step, which can be done using the past feedback to deduce the value of x_{i-1}, and hence, recursively, the value of u_{i-1}.

Now we prove by induction that these two encoding algorithms can achieve the same square dividing strategy we described in the full feedback case. At the first step, this is obvious. Generally, we need to prove for the two clusters studied in the proof of Theorem 1, the new strategies give precisely the desired results. In the case of the first cluster of size $a_k \times b_k$, the two encoders are sending 1's for the bigger subsets of sizes a_{k-1} and b_{k-1}, respectively, and 0's for the smaller ones. It easy to check that the resulting outputs, shown in Figure 7a, are precisely the ones we need in the proof of the Theorem 1. For the second cluster, $a_{k-1} \times b_{k-2} \vee a_{k-2} \times b_{k-1}$, the channel inputs calculated by the two encoders in accordance with the above prescriptions are shown in Figure 7b ; note that the resulting outputs again exactly satisfy the requirements of the proof of the Theorem 1. The next step, shown in Figure 7c, has the (5,3)-code of Figure 4 embedded within it. We omit the general step in the induction argument because its validity should be apparent by now.

(c) The inner bound to the zero error capacity region

The three families of zero-error codes we have introduced can be combined by tangent lines representing time sharing to produce an inner bound to the zero-error capacity region of the binary adder channel with partial feedback. This bound can be mildly improved in the low rate region for the uninformed encoder by appealing to an inner bound to the zero-error capacity region derived by Kasami et al. [4] for the case in which there is no feedback to either encoder; clearly, any inner bound for that case is an inner bound for the partial feedback case. That bound and a time sharing line joining it to the second of our code families completes our overall inner bound depicted in Figure 2. The straight-line portion of this bound between the curve of our first code family and the Fibonacci code has slope -1 and a rate sum of 1.3885.

4. CODES GENERATED BY DIFFERENCE EQUATIONS FOR THE FULL FEEDBACK CASE

(a) Refinement of the Fibonacci code

We call a k-attainable pair (a, b) optimal, if neither $(a+1, b)$ nor $(a, b+1)$ is k-attainable. Consider the Fibonacci code sizes :

$$(2,1), (3,2), (5,3), (8,5), (13,8), (21,13), \cdots$$

It is not hard to prove that the first 3 terms are optimal. It turns out, however, that (9,5) is 4-attainable and (14,8) is 5-attainable. This suggests that there may exist code generating equations which generate codes with asymptotically equal rates greater than R_f. In this regard it can be shown that the following equations are code generating for the full feedback case:

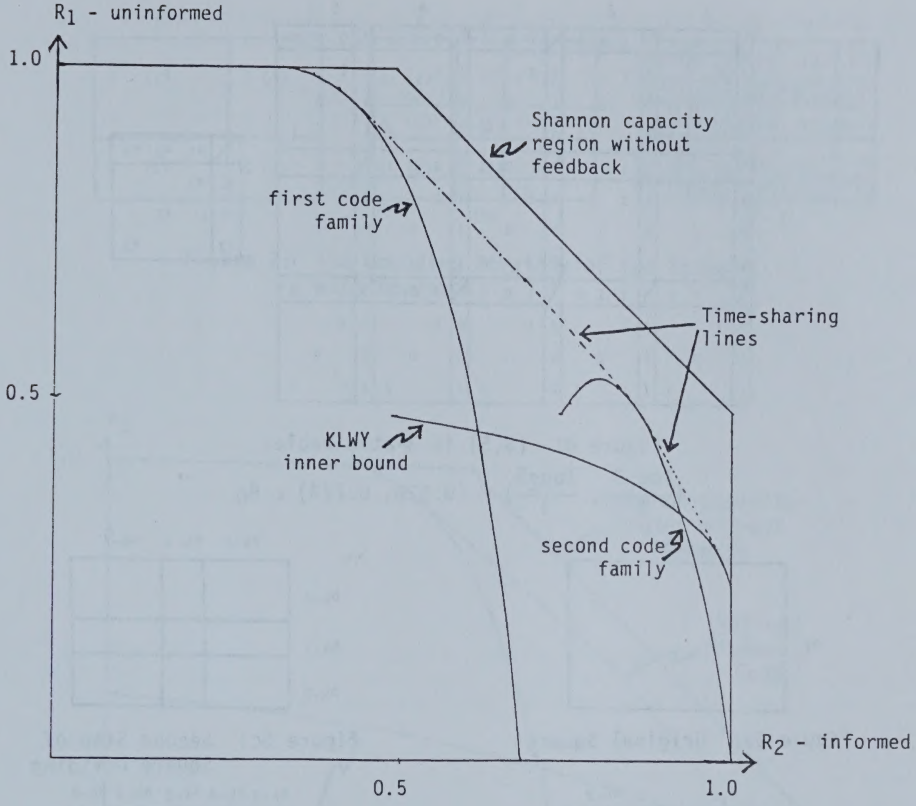


Figure 2: Rates of Code Families and Inner Bound to Capacity Region for Partial Feedback Case.

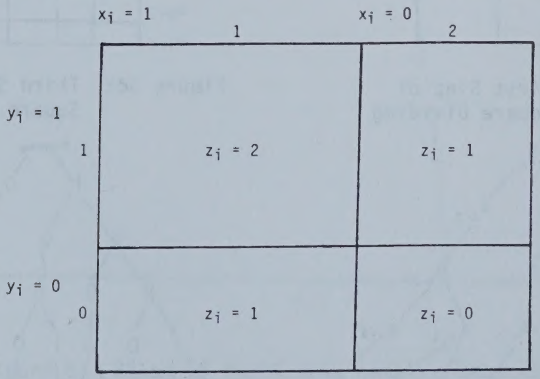

$x_i = 1$
$x_i = 0$
1
2
$y_i = 1$
1
$z_i = 2$
$z_i = 1$
$y_i = 0$
0
$z_i = 1$
$z_i = 0$

Figure 3: Square Dividing Strategy

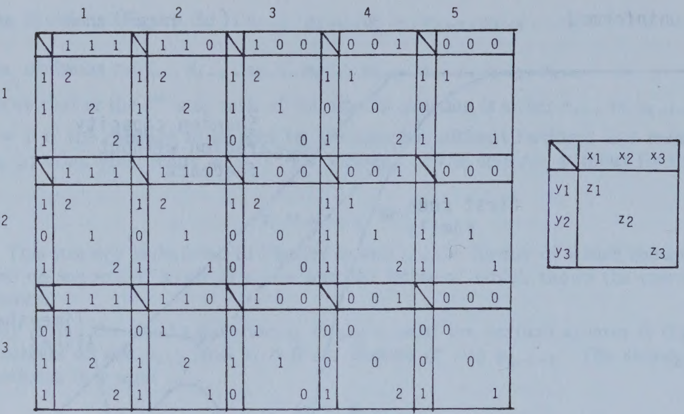

1 2 3 4 5
1 1 1 | 1 1 0 | 1 0 0 | 0 0 1 | 0 0 0
1
1 2 | 1 2 | 1 2 | 1 1 | 1 1
1 2 | 1 2 | 1 1 | 0 0 | 0 0
1 2 | 1 1 | 0 0 | 1 2 | 1 1
2
1 2 | 1 2 | 1 2 | 1 1 | 1 1
0 1 | 0 1 | 0 0 | 1 1 | 1 1
1 2 | 1 1 | 0 0 | 1 2 | 1 1
3
0 1 | 0 1 | 0 1 | 0 0 | 0 0
1 2 | 1 2 | 1 1 | 0 0 | 0 0
1 2 | 1 1 | 0 0 | 1 2 | 1 1
x1 x2 x3
y1 z1
y2 z2
y3 z3

Figure 4: (3,5) is 3-attainable:

$$\left(\frac{\log_2 3}{3}, \frac{\log_2 5}{3}\right) = (0.528, 0.774) \in R_0$$

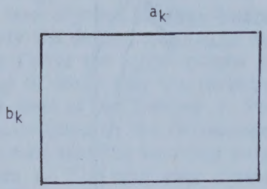


Figure 5a: Original Square

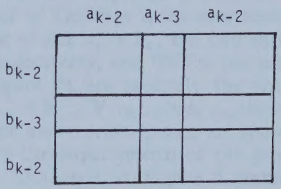

a_{k-2} a_{k-3} a_{k-2}
b_{k-2}
b_{k-3}
b_{k-2}

Figure 5c: Second Step of Square Dividing

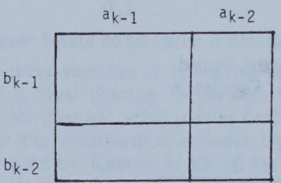

a_{k-1} a_{k-2}
b_{k-1}
b_{k-2}

Figure 5b: First Step of Square Dividing

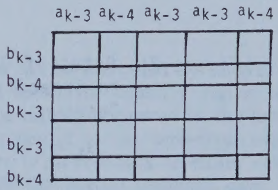

a_{k-3} a_{k-4} a_{k-3} a_{k-3} a_{k-4}
b_{k-3}
b_{k-4}
b_{k-3}
b_{k-3}
b_{k-4}

Figure 5d: Third Step of Square Dividing

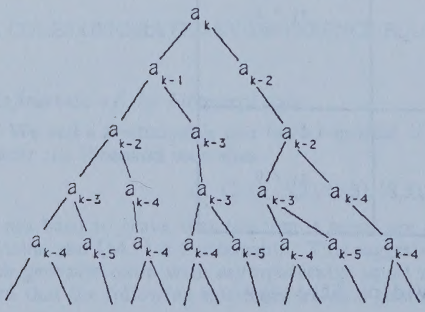

a_k
a_{k-1} a_{k-2}
a_{k-2} a_{k-3} a_{k-2}
a_{k-3} a_{k-4} a_{k-3} a_{k-3} a_{k-4}
a_{k-4} a_{k-5} a_{k-4} a_{k-4} a_{k-5} a_{k-4} a_{k-5} a_{k-4}

Figure 6a: Square Dividing Tree

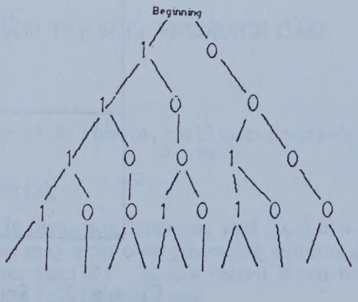
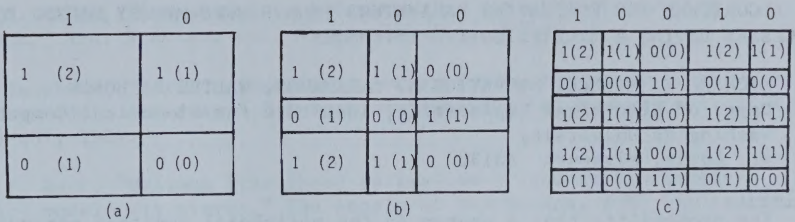

Beginning
1 0
1 0 0
1 0 0 1 0
1 0 0 1 0 1 0 0

Figure 6b: Encoding Tree of the Encoder without Feedback



1	0
1 (2)	1 (1)
0 (1)	0 (0)

(a)

1	0	0
1 (2)	1 (1)	0 (0)
0 (1)	0 (0)	1 (1)
1 (2)	1 (1)	0 (0)

(b)

1	0	0	1	0
1(2)	1(1)	0(0)	1(2)	1(1)
0(1)	0(0)	1(1)	0(1)	0(0)
1(2)	1(1)	0(0)	1(2)	1(1)
1(2)	1(1)	0(0)	1(2)	1(1)
0(1)	0(0)	1(1)	0(1)	0(0)

(c)

Figure 7: The Encoding Stratety of the Encoder with Feedback

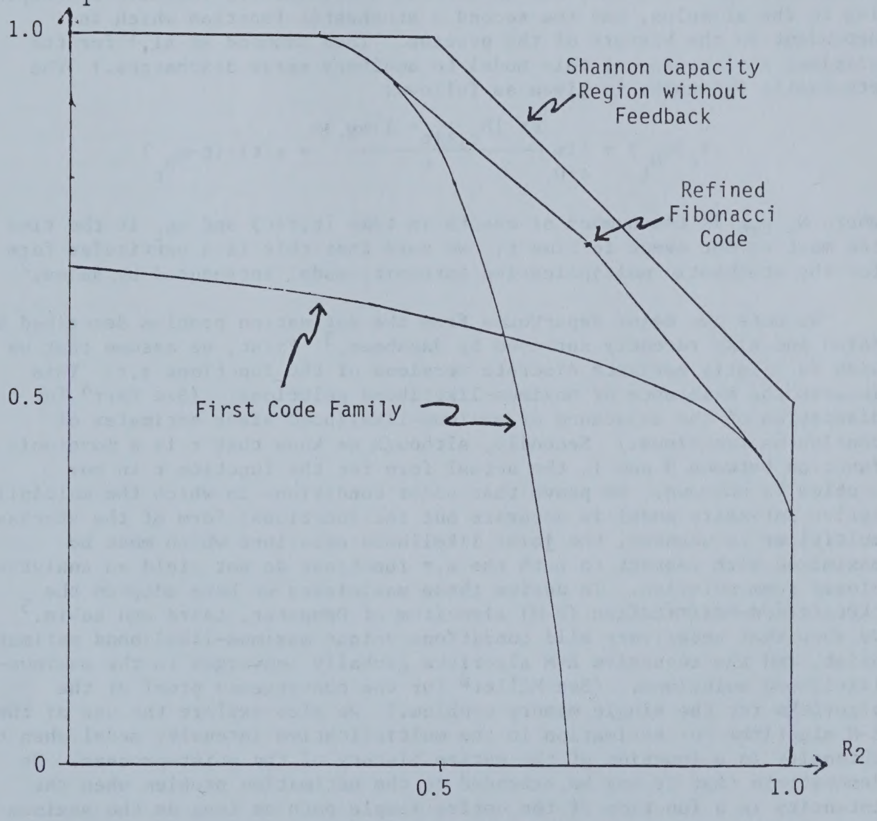


Figure 8: Rates of First Code Family, Extended Fibonacci Code, and Inner Bound to Capacity Region for Full Feedback Case

E-M ALGORITHMS FOR ESTIMATING PARAMETERS FROM SINGLE-MEMORY MARKOV POINT-PROCESSES HAVING A MULTIPLICATIVE INTENSITY

MICHAEL I. MILLER, NEOPHYTOS A. KARAMANOS, WALTER R. BOSCH
Dept. of Electrical Engineering, Institute for Biomedical Computing
Washington University
St. Louis, Missouri 63130

ABSTRACT

The probability that a neuron in the peripheral auditory nervous system discharges during some specified time interval depends on both a stimulus function and on effects due to earlier discharge events. With the objective of estimating parameters related to the stimulus function which are free from the distortion introduced by effects due to previous discharges, a maximum-likelihood estimation scheme has been developed. We have chosen a multiplicative intensity model to describe auditory-nerve discharges in which the intensity of the process is stochastic and the product of two functions; one an unknown deterministic function corresponding to the stimulus, and the second a stochastic function which is dependent on the history of the process. (See Gaumond et al.[1] for the original application of this model to auditory nerve discharges.) The stochastic intensity is given as follows:

$$\lambda_t(\omega_{N_t}) = \lim_{\varepsilon\to 0} \frac{\Pr\,[N_{t,t+\varepsilon} = 1 | \omega_{N_t}]}{\varepsilon} = s(t)r(t-\omega_{N_t})$$

where $N_{t,t+\varepsilon}$ is the number of events in time $[t,t+\varepsilon)$ and ω_{N_t} is the time of the most recent event to time t. We note that this is a particular form for the stochastic multiplicative intensity model introduced by Aalen.[2]

We note two major departures from the estimation problem described by Aalen and also recently surveyed by Jacobsen.[3] First, we assume that we wish to jointly estimate discrete versions of the functions s,r. This insures the existence of maximum-likelihood solutions. (See Karr[4] for a discussion of the existence of maximum-likelihood sieve estimates of continuous functions.) Secondly, although we know that r is a monotonic function between 0 and 1, the actual form for the function r in our problem is unknown. We prove that under conditions in which the multiplicative intensity model is accurate but the functional form of the stochastic multiplier is unknown, the joint likelihood equations which must be maximized with respect to both the s,r functions do not yield an analytic closed form solution. To derive these maximizers we have adopted the *expectation-maximization* (E-M) algorithm of Dempster, Laird and Rubin.[5] We show that under very mild conditions unique maximum-likelihood estimates exist, and the recursive E-M algorithm globally converges to the maximum-likelihood solutions. (See Miller[6] for the convergence proof of the algorithm for the single memory problem.) We also explore the use of the E-M algorithm for estimation in the multiplicative intensity model when the intensity is a function of the entire history of the point-process. We demonstrate that it may be extended to the estimation problem when the intensity is a function of the entire sample path as long as the maximum value of the stochastic multiplier r is known.

References

1. R. P. Gaumond, D. O. Kim, and C. E. Molnar, "Response of cochlear nerve fibers to brief acoustic stimuli: role of discharge history effects," J. Acoust. Soc. Am., vol. 74, pp. 1392-1398, 1983.

2. O. O. Aalen, "Nonparametric inference for a family of counting processes," Ann. Statis., vol. 6, pp. 701-726, 1978.

3. M. Jacobsen, "Maximum likelihood estimation in the multiplicative intensity model: a survey," International Statistical Review, vol. 52.2, pp. 193-207, 1984.

4. A. F. Karr, "Maximum likelihood estimation of the multiplicative intensity model, via sieves," The Annals of Statistics, 1985 (in review).

5. A. D. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," J. of the Royal Statistical Society, vol. B.39, pp. 1-37, 1977.

6. M. I. Miller, "Algorithms for removing recovery-related distortion from auditory nerve discharge patterns," J. Acoust. Soc. Am., vol. 77 (4), pp. 1452-1464, 1985.

ESTIMATION OF THE RATE OF A DISCRETE-TIME MULTIVARIATE POINT PROCESS

JOHN GUBNER and PRAKASH NARAYAN[1]
Electrical Engineering Department
University of Maryland
College Park, Maryland 20742

ABSTRACT

We introduce the notion of a discrete-time, multivariate point process which can arise in the modeling of an optical communication system. We wish to estimate the rate of this process at time t given the past of the process up to time $t-1$. This requires the computation of a certain conditional expectation; we perform this computation by introducing an absolutely continuous change of measure and then applying the generalized Bayes' rule.

I. INTRODUCTION

Suppose that a laser beam, whose intensity is modulated by an information source, strikes a photodetector which is a part of the receiver of an optical communication system. Suppose that the beam is also influenced by external random phenomena; this occurs, for example, if the beam passes through a turbulent atmosphere [1]. The occurrence of photoelectrons at the detector can then be modeled as a doubly-stochastic, time-space Poisson process [1]. It is often necessary to compute the conditional expectation of the rate of such a process, for example, in performing a likelihood ratio test (Snyder [2], Chapter 2 and Chapter 6). In general, the computation of this conditional expectation is quite difficult, although results are available if the rate process has a Gaussian form, and the photodetector surface is very large; in addition, results for *linear* estimates are available when the photodetector surface is arbitrary, but the rate process still has a Gaussian form [3]. (see also [4, 5] for related filtering results).

Let D represent the photosensitive surface of the photodetector. It is well known that the probability is negligible that more than one photoelectron will occur in the entire region D during a time interval of length Δt, if Δt is sufficiently small. Let $D_1, \ldots, D_K$, be a partition of D into disjoint subregions. Let Δt be "sufficiently small," and let $n_t(k)$ denote the number of photoelectrons occurring in the region D_k, during an interval $(t, t+\Delta t]$. We should expect that for each $1 \leq k \leq K$, $n_t(k)$ takes only the values 0 or 1. In addition, we can have $n_t(k) = 1$ for at most one k; for $i \neq k$, we must have $n_t(i) = 0$. In this paper we formulate the discrete-time version of the rate-estimation problem by introducing the notion of a discrete-time, multivariate point process. By adapting the continuous-time procedures found in (Bremaud [6], Chapter IV; see also pp. 69-70, 80 of Chapter III), we solve the problem of estimating the rate of a discrete-time, multivariate point process. (For more general discrete-time procedures, see [7, 8]).

---

[1] This research was sponsored by the Office of Naval Research under grant no.N0001485-G-0102 and by the Minta Martin Fund for Aerospace Research from the University of Maryland at College Park.

II. PROBLEM STATEMENT

Let $K \geq 1$ be a fixed integer. Let $(\Omega, \mathbb{F}, \mathbf{P})$ be a probability space on which all the random variables in this section are defined. We call $\{ n_t(k);\ t \geq 1,\ 1 \leq k \leq K \}$ a discrete-time, multivariate point process if each $n_t(k)$ takes only the values 0 and 1 and if the events $\{n_t(k) = 1\}$, $1 \leq k \leq K$, are disjoint, so that simultaneous events do not occur. Next, let $\mathbb{G}_0$ denote the trivial σ-field on Ω, and set

$$\mathbb{G}_t \triangleq \sigma\{ n_s(k);\ 1 \leq s \leq t,\ 1 \leq k \leq K \}.$$

Now, let $\mathbb{X}$ be a sub-σ-field of $\mathbb{F}$, and let

$$\mathbb{F}_t \triangleq \mathbb{G}_t \vee \mathbb{X} \ ; \quad t \geq 0,$$

denote the smallest σ-field containing $\mathbb{G}_t \bigcup \mathbb{X}$. We assume that the random variables

$$\lambda_t(k) \triangleq \mathbf{E} [n_t(k) \mid \mathbb{F}_{t-1}] \ ; \quad t \geq 1,$$

are actually $\mathbb{X}$-measurable (note that $\mathbb{X} = \mathbb{F}_0$) with *known* joint distributions. For example, we might take $\mathbb{X} = \sigma(X)$ for some random variable X with known distribution function $F(x)$. Then each $\lambda_t(k)$ would be some Borel function of X. Clearly, the joint distributions of the $\{ \lambda_t(k);\ t \geq 1,\ 1 \leq k \leq K \}$ would be known, at least in principle. Now, observe that $\lambda_t(k) = \mathbf{P}(n_t(k) = 1 \mid \mathbb{F}_{t-1})$; since we assume that simultaneous events do not occur,

$$\sum_{k=1}^{K} \lambda_t(k) = \mathbf{P}\left(\bigcup_{k=1}^{K} \{ n_t(k) = 1 \} \mid \mathbb{F}_{t-1} \right).$$

This implies that $p_t \triangleq 1 - \sum_{k=1}^{K} \lambda_t(k)$ is also a conditional probability, and hence, nonnegative and bounded above by 1.

Our objective is to compute

$$\hat{\lambda}_t(k) \triangleq \mathbf{E} [n_t(k) \mid \mathbb{G}_{t-1}]$$

in terms of the known joint distribution of $\{ \lambda_s(k);\ 1 \leq s \leq t,\ 1 \leq k \leq K \}$.

III. A REFORMULATION OF THE PROBLEM

We shall solve our problem by reformulating the probabilistic setting above on a different probability space, $(\tilde{\Omega}, \tilde{\mathbb{F}}, \mathbf{Q})$. On $(\tilde{\Omega}, \tilde{\mathbb{F}}, \mathbf{Q})$ let $\{ \nu_t(k);\ t \geq 1,\ 1 \leq k \leq K \}$ be a discrete-time, multivariate point process. In a manner analogous to that outlined in the previous section, we take $\tilde{\mathbb{G}}_0$ to be the trivial σ-field on $\tilde{\Omega}$, and set

$$\tilde{\mathbb{G}}_t \triangleq \sigma\{ \nu_s(k);\ 1 \leq s \leq t,\ 1 \leq k \leq K \}\ ;\quad t \geq 1.$$

Next, let $\tilde{\mathbb{X}}$ be a sub-σ-field of $\tilde{\mathbb{F}}$, and set

$$\tilde{\mathbb{F}}_t \triangleq \tilde{\mathbb{G}}_t \vee \tilde{\mathbb{X}}\ ;\quad t \geq 0.$$

We now assume that

$$\mathbf{E}_{\mathbf{Q}} [\, \nu_t(k) \mid \tilde{\mathbb{F}}_{t-1} \,] = \mu_t(k), \tag{1}$$

where the $\{ \mu_t(k) \}$ are arbitrary constants satisfying $\mu_t(k) > 0$ and $q_t \triangleq 1 - \sum_{k=1}^{K} \mu_t(k) > 0$. (The symbol $\mathbf{E}_{\mathbf{Q}}$ denotes expectation with respect to the measure $\mathbf{Q}$). As a consequence of the above assumption, under the measure $\mathbf{Q}$, each $\nu_t(k)$ is independent of $\tilde{\mathbb{G}}_{t-1}$. In addition, the σ-field $\tilde{\mathbb{G}}_t$ is independent of the σ-field $\tilde{\mathbb{X}}$ (see Appendix). Next, let the random variables $\{ \tilde{\lambda}_t(k);\ t \geq 1,\ 1 \leq k \leq K \}$ defined on $\tilde{\Omega}$ be $\tilde{\mathbb{X}}$-measurable *and* have the *same* joint distributions under $\mathbf{Q}$ as $\{ \lambda_t(k);\ t \geq 1,\ 1 \leq k \leq K \}$ (defined on Ω) under $\mathbf{P}$.

Given the preceding probabilistic setting on $(\tilde{\Omega}, \tilde{\mathbb{F}}, \mathbf{Q})$, we make the following definitions. Let

$$\tilde{p}_t \triangleq 1 - \sum_{k=1}^{K} \tilde{\lambda}_t(k)\ ;\quad t \geq 1,$$

$$l_t \triangleq \sum_{k=1}^{K} \left(\frac{\tilde{\lambda}_t(k)}{\mu_t(k)} - \frac{\tilde{p}_t}{q_t} \right) \nu_t(k) + \frac{\tilde{p}_t}{q_t}\ ;\quad t \geq 1, \tag{2}$$

and

$$L_t \triangleq \prod_{s=1}^{t} l_s \; ; \quad t \geq 1,$$

with $L_0 \equiv 1$. Observe that the denominators in (2) are nonzero and deterministic. Also, $0 \leq \tilde{\lambda}_t(k) \leq 1$ $\mathbf{Q}$-a.s., and hence, $\tilde{p}_t$ is clearly bounded and $\mathbf{Q}$-integrable. (Recall that the joint distributions of the $\{ \tilde{\lambda}_t(k) \}$ under $\mathbf{Q}$ are the same as those of the $\{ \lambda_t(k) \}$ under $\mathbf{P}$). Consequently, l_t and L_t are $\mathbf{Q}$-integrable. Since $\sum_{k=1}^{K} \tilde{\lambda}_t(k) = 1 - \tilde{p}_t$, and $\sum_{k=1}^{K} \mu_t(k) = 1 - q_t$, it is easy to see that $\mathbf{E}_{\mathbf{Q}} [\, l_t \mid \tilde{\mathbb{F}}_{t-1} \,] = 1$. Since $L_t = L_{t-1} l_t$, it is clear that

$$\mathbf{E}_{\mathbf{Q}} [\, L_t \mid \tilde{\mathbb{F}}_{t-1} \,] = L_{t-1} \; ; \tag{3}$$

i.e., L_t is an $\tilde{\mathbb{F}}_t$-martingale under $\mathbf{Q}$. Since, $\mathbf{E}_{\mathbf{Q}} [\, L_t \,] = \mathbf{E}_{\mathbf{Q}} [\, L_1 \,] = \mathbf{E}_{\mathbf{Q}} [\, l_1 \,] = 1$, we can define a new measure $\tilde{\mathbf{P}}$ on $\tilde{\mathbb{F}}_t$ by

$$\tilde{\mathbf{P}}(F) \triangleq \int_F L_t \, d\mathbf{Q} \; ; \quad F \in \tilde{\mathbb{F}}_t \, .$$

(Technically, we should show that the family $\{ L_t \}$ is uniformly integrable. However, for our purposes this is not necessary since if we wish to compute $\mathbf{E} [\, n_\tau(k) \mid \mathbb{G}_{\tau-1} \,]$ for some τ, we can select any finite $T \geq \tau$ and then restrict our attention to $1 \leq t \leq T$). Observe that since $L_0 = 1$, $\tilde{\mathbf{P}} = \mathbf{Q}$ on $\tilde{\mathbb{X}}$. If $\tilde{\mathbf{E}}$ denotes expectation with respect to the measure $\tilde{\mathbf{P}}$, it is not hard to verify (since simultaneous events do not occur) that

$$\begin{aligned} \tilde{\mathbf{E}} [\, \nu_t(k) \mid \tilde{\mathbb{F}}_{t-1} \,] &= \mathbf{E}_{\mathbf{Q}} [\, \nu_t(k) \frac{L_t}{L_{t-1}} \mid \tilde{\mathbb{F}}_{t-1} \,] \\ &= \mathbf{E}_{\mathbf{Q}} [\, \nu_t(k) \, l_t \mid \tilde{\mathbb{F}}_{t-1} \,] \\ &= \tilde{\lambda}_t(k), \end{aligned} \tag{4}$$

where the first equality in (4) follows from the generalized Bayes' rule. Now, since $\tilde{\mathbf{P}} = \mathbf{Q}$ on $\tilde{\mathbb{X}}$,

$$\tilde{\lambda}_t(k) = \tilde{\mathbf{E}} [\, \nu_t(k) \mid \tilde{\mathbb{F}}_{t-1} \,]$$

under $\tilde{\mathbf{P}}$ is probabilistically equivalent to $\lambda_t(k)$ under $\mathbf{P}$. In fact, we can make the following statement. The probabilistic relationships among

$$\{ n_t(k) \},\ \{ \lambda_t(k) \},\ \{ \mathbb{F}_t \},\ \{ \mathbb{G}_t \},\ \text{and}\ \boldsymbol{X}$$

under $\mathbf{P}$ are the same as those among

$$\{ \nu_t(k) \},\ \{ \tilde{\lambda}_t(k) \},\ \{ \tilde{\mathbb{F}}_t \},\ \{ \tilde{\mathbb{G}}_t \},\ \text{and}\ \tilde{\boldsymbol{X}}$$

under $\tilde{\mathbf{P}}$. In the next section we shall compute $\tilde{\mathbf{E}}\,[\, \nu_t(k) \mid \tilde{\mathbb{G}}_{t-1} \,]$ explicitly as a Borel function of $\{ \nu_s(i);\ 1 \le s \le t-1,\ 1 \le i \le K \}$. From the statement above, it follows that $\mathbf{E}\,[\, n_t(k) \mid \mathbb{G}_{t-1} \,]$ will be equal to the same Borel function applied to $\{ n_s(i);\ 1 \le s \le t-1,\ 1 \le i \le K \}$.

IV. CALCULATIONS

In this section we compute $\tilde{\mathbf{E}}\,[\, \nu_t(k) \mid \tilde{\mathbb{G}}_{t-1} \,]$. By Bayes' rule,

$$\tilde{\mathbf{E}}\,[\, \nu_t(k) \mid \tilde{\mathbb{G}}_{t-1} \,] = \frac{\mathbf{E}_{\mathrm{Q}}\,[\, \nu_t(k)\, L_t \mid \tilde{\mathbb{G}}_{t-1} \,]}{\mathbf{E}_{\mathrm{Q}}\,[\, L_t \mid \tilde{\mathbb{G}}_{t-1} \,]}\,. \tag{5}$$

Next, observe that

$$\mathbf{E}_{\mathrm{Q}}\,[\, L_t \mid \tilde{\mathbb{G}}_{t-1} \,] = \mathbf{E}_{\mathrm{Q}}\,[\, \mathbf{E}_{\mathrm{Q}}\,[\, L_t \mid \tilde{\mathbb{F}}_{t-1} \,] \mid \tilde{\mathbb{G}}_{t-1} \,] = \mathbf{E}_{\mathrm{Q}}\,[\, L_{t-1} \mid \tilde{\mathbb{G}}_{t-1} \,]$$

by equation (3). In addition,

$$\begin{aligned} \mathbf{E}_{\mathrm{Q}}\,[\, \nu_t(k)\, L_t \mid \tilde{\mathbb{G}}_{t-1} \,] &= \mathbf{E}_{\mathrm{Q}}\,[\, \mathbf{E}_{\mathrm{Q}}\,[\, \nu_t(k)\, L_t \mid \tilde{\mathbb{G}}_t \,] \mid \tilde{\mathbb{G}}_{t-1} \,] \\ &= \mathbf{E}_{\mathrm{Q}}\,[\, \nu_t(k)\, \mathbf{E}_{\mathrm{Q}}\,[\, L_t \mid \tilde{\mathbb{G}}_t \,] \mid \tilde{\mathbb{G}}_{t-1} \,]. \end{aligned}$$

Hence, (5) can be rewritten as

$$\tilde{\mathbf{E}}\,[\, \nu_t(k) \mid \tilde{\mathbb{G}}_{t-1} \,] = \frac{\mathbf{E}_{\mathrm{Q}}\,[\, \nu_t(k)\, \mathbf{E}_{\mathrm{Q}}\,[\, L_t \mid \tilde{\mathbb{G}}_t \,] \mid \tilde{\mathbb{G}}_{t-1} \,]}{\mathbf{E}_{\mathrm{Q}}\,[\, L_{t-1} \mid \tilde{\mathbb{G}}_{t-1} \,]}\,. \tag{6}$$

Before proceeding further, it will be convenient to introduce the following notation. For each $t \geq 1$, let

$$\nu_t \triangleq [\, \nu_t(1), \ldots, \nu_t(K) \,]' ,$$

where $'$ denotes transpose. Then ν_t is a K-dimensional random vector. Now, let

$$\overline{\nu}_t \triangleq [\, \nu_1, \ldots, \nu_t \,]' .$$

Clearly, $\overline{\nu}_t$ is a tK-dimensional random vector. Next, let

$$z_t \triangleq [\, z_t(1), \ldots, z_t(K) \,]' \quad \text{and} \quad \overline{z}_t \triangleq [\, z_1, \ldots, z_t \,]'$$

denote "dummy" variables in $\mathbb{R}^K$ and $\mathbb{R}^{tK}$, respectively. Define

$$h_t(z_t) = \sum_{k=1}^{K} \left\{ \frac{\lambda_t(k)}{\mu_t(k)} - \frac{p_t}{q_t} \right\} z_t(k) + \frac{p_t}{q_t}, \tag{7}$$

and

$$\tilde{h}_t(z_t) = \sum_{k=1}^{K} \left\{ \frac{\tilde{\lambda}_t(k)}{\mu_t(k)} - \frac{\tilde{p}_t}{q_t} \right\} z_t(k) + \frac{\tilde{p}_t}{q_t} .$$

Clearly, $h_t(z_t)$ is an $\mathbb{X}$-measurable random variable and $\tilde{h}_t(z_t)$ is an $\tilde{\mathbb{X}}$-measurable random variable. In fact, $\tilde{h}_t(z_t)$ under $\mathbf{Q}$ is probabilistically equivalent to $h_t(z_t)$ under $\mathbf{P}$. It is then clear that the same can be said of

$$H_t(\overline{z}_t) \triangleq \prod_{s=1}^{t} h_s(z_s), \tag{8}$$

and

$$\tilde{H}_t(\overline{z}_t) \triangleq \prod_{s=1}^{t} \tilde{h}_s(z_s).$$

Next observe that $l_t = \tilde{h}_t(\nu_t)$, and that $L_t = \tilde{H}_t(\overline{\nu}_t)$. We are now ready to compute $\mathbf{E}_{\mathrm{Q}}[\, L_t \mid \tilde{\mathbb{G}}_t \,]$ in (6) for each $t \geq 1$. (Note that $\mathbf{E}_{\mathrm{Q}}[\, L_0 \mid \tilde{\mathbb{G}}_0 \,] = \mathbf{E}_{\mathrm{Q}}[\, L_0 \,] = 1$). Observe that $L_t = \tilde{H}_t(\overline{\nu}_t)$ is an $\tilde{\mathbb{X}}$-measurable function of a $\tilde{\mathbb{G}}_t$-measurable random vector. Since $\tilde{\mathbb{X}}$ and $\tilde{\mathbb{G}}_t$ are independent under $\mathbf{Q}$, it follows that

$$\mathbf{E}_{\mathbf{Q}}\,[\,\tilde{H}_t(\overline{\nu}_t)\ \mid\ \tilde{\mathcal{G}}_t\,] = \mathbf{E}_{\mathbf{Q}}\,[\,\tilde{H}_t(\overline{z}_t)\,]\,\big|_{\,\overline{z}_t = \overline{\nu}_t}.$$

From the remarks preceding equation (8), we have

$$\mathbf{E}_{\mathbf{Q}}\,[\,L_t\ \mid\ \tilde{\mathcal{G}}_t\,] = \mathbf{E}\,[\,H_t(\overline{z}_t)\,]\,\big|_{\,\overline{z}_t = \overline{\nu}_t}.$$

We now set

$$f_t(\overline{z}_t) \triangleq \mathbf{E}\,[\,H_t(\overline{z}_t)\,]. \tag{9}$$

Note that $f_t(\overline{z}_t)$ is a deterministic function of $\overline{z}_t$. Equation (6) becomes

$$\tilde{\mathbf{E}}\,[\,\nu_t(k)\ \mid\ \tilde{\mathcal{G}}_{t-1}\,] = \frac{\mathbf{E}_{\mathbf{Q}}\,[\,\nu_t(k)\,f_t(\overline{\nu}_t)\ \mid\ \tilde{\mathcal{G}}_{t-1}\,]}{f_{t-1}(\overline{\nu}_{t-1})}.$$

We can write

$$\mathbf{E}_{\mathbf{Q}}\,[\,\nu_t(k)\,f_t(\overline{\nu}_t)\ \mid\ \tilde{\mathcal{G}}_{t-1}\,] = \mathbf{E}_{\mathbf{Q}}\,[\,\nu_t(k)\,f_t(\overline{\nu}_{t-1},\nu_t)\ \mid\ \tilde{\mathcal{G}}_{t-1}\,].$$

Now, equation (1) implies that ν_t is independent of $\mathcal{G}_{t-1}$ under $\mathbf{Q}$. Therefore,

$$\mathbf{E}_{\mathbf{Q}}\,[\,\nu_t(k)\,f_t(\overline{\nu}_t)\ \mid\ \tilde{\mathcal{G}}_{t-1}\,] = \mathbf{E}_{\mathbf{Q}}\,[\,\nu_t(k)\,f_t(\overline{z}_{t-1},\nu_t)\,]\,\big|_{\,\overline{z}_{t-1} = \overline{\nu}_{t-1}}.$$

Since simultaneous events do not occur, it is trivial to compute

$$\begin{aligned}\mathbf{E}_{\mathbf{Q}}\,[\,\nu_t(k)\,f_t(\overline{z}_{t-1},\nu_t)\,] &= f_t(\overline{z}_{t-1},e_k)\mathbf{Q}(\,\nu_t(k) = 1\,)\\ &= \mu_t(k)\,f_t(\overline{z}_{t-1},\,e_k),\end{aligned}$$

where e_k is the standard unit vector in $\mathbb{R}^K$ with a 1 in the kth position and a 0 in the other $K-1$ positions. We conclude that

$$\tilde{\mathbf{E}}\,[\,\nu_t(k)\ \mid\ \tilde{\mathcal{G}}_{t-1}\,] = \frac{\mu_t(k)f_t(\overline{\nu}_{t-1},\,e_k)}{f_{t-1}(\overline{\nu}_{t-1})},$$

and hence,

$$\mathbf{E}\,[\, n_t(k) \mid \mathcal{G}_{t-1} \,] = \frac{\mu_t(k) f_t(\overline{n}_{t-1}, e_k)}{f_{t-1}(\overline{n}_{t-1})},$$

with the obvious meaning of the symbol $\overline{n}_{t-1}$.

V. SUMMARY

We have shown that if $\{\, n_t(k);\ t \geq 1,\ 1 \leq k \leq K \,\}$ is a discrete-time, multivariate point process residing in the probabilistic setting outlined in Section II, then

$$\mathbf{E}\,[\, n_t(k) \mid \mathcal{G}_{t-1} \,] = \frac{\mu_t(k) f_t(\overline{n}_{t-1}, e_k)}{f_{t-1}(\overline{n}_{t-1})},$$

where $f_t(\overline{z}_t)$ is given by equations (7), (8), and (9).

The $\mu_t(k)$ introduced in Section III were arbitrary; if we set $\mu_t(k) = \dfrac{1}{K+1}$ for all t and all k in the preceding equations, we find that

$$\mathbf{E}\,[\, n_t(k) \mid \mathcal{G}_{t-1} \,] = \frac{a_t(\overline{n}_{t-1}, e_k)}{a_{t-1}(\overline{n}_{t-1})},$$

where

$$a_t(\overline{z}_t) \triangleq \mathbf{E}\,[\, \prod_{s=1}^{t} \left\{ \sum_{k=1}^{K} (\lambda_s(k) - p_s) n_s(k) + p_s \right\} \,].$$

APPENDIX

For $1 \leq s \leq t$, let z_s denote either the zero vector in $\mathbb{R}^K$ or any standard unit vector in $\mathbb{R}^K$. In this appendix we prove that if $E \in \mathcal{X}$, then (using the notation of Sections III and IV)

$$\mathrm{Q}(\, \nu_t = z_t, \ldots, \nu_1 = z_1, E \,) = \left\{ \prod_{s=1}^{t} \mathrm{Q}(\, \nu_s = z_s \,) \right\} \mathrm{Q}(E).$$

Since $\mathcal{G}_t = \sigma\{\, \nu_1, \ldots, \nu_t \,\}$, this will prove that $\mathcal{G}_t$ and $\mathcal{X}$ are independent under Q.

Proof. Using the definition of conditional probability,

$$Q(\nu_t = z_t, \ldots, \nu_1 = z_1, E) = \int_{\{\nu_{t-1} = z_{t-1}, \ldots, \nu_1 = z_1, E\}} Q(\nu_t = z_t \mid \tilde{\mathbb{F}}_{t-1})\, dQ. \tag{10}$$

Since simultaneous events do not occur, if $z_t = e_k$, $Q(\nu_t = z_t \mid \tilde{\mathbb{F}}_{t-1}) = \mu_t(k)$. If $z_t = 0$, $Q(\nu_t = z_t \mid \tilde{\mathbb{F}}_{t-1}) = 1 - \sum_{k=1}^{K} \mu_t(k)$. Since $Q(\nu_t = z_t \mid \tilde{\mathbb{F}}_{t-1})$ is deterministic,

$$Q(\nu_t = z_t \mid \tilde{\mathbb{F}}_{t-1}) = Q(\nu_t = z_t).$$

Hence (10) becomes

$$Q(\nu_t = z_t, \ldots, \nu_1 = z_1, E) = Q(\nu_t = z_t) Q(\nu_{t-1} = z_{t-1}, \ldots, \nu_1 = z_1, E).$$

The remainder of the proof by induction is clear.

QED

ACKNOWLEDGEMENT

It is a pleasure to thank Professor A. M. Makowski for several illuminating discussions on the applications of an absolutely continuous change of measure to filtering problems.

REFERENCES

[1] D. L. Snyder, "Applications of Stochastic Calculus for Point Process Models Arising in Optical Communication," pp. 789-804 in *Communication Systems and Random Process Theory*, ed. J. K. Skwirzynski, Sijthoff and Noordhoff, Alphen aan der Rijn, The Netherlands (1978).

[2] D. L. Snyder, *Random Point Processes*, John Wiley & Sons, New York (1975).

[3] J. Gubner and P. Narayan, "Estimation of the Rate of a Doubly-Stochastic Time-Space Poisson Process," *Systems & Control Letters*, Submitted, (1985).

[4] D. L. Snyder and P. M. Fishman, "How to Track a Swarm of Fireflies by Observing Their Flashes," *IEEE Transactions on Information Theory*, vol. IT-21, pp. 692-695 (Nov. 1975).

[5] I. B. Rhodes and D. L. Snyder, "Estimation and Control Performance for Space-Time

Point-Process Observations," *IEEE Transactions on Automatic Control,* vol. AC-22, No. 3, pp. 338-346 (June 1977).

[6] P. Bremaud, *Point Process and Queues, Martingale Dynamics,* Springer-Verlag, New York (1981).

[7] P. M. Bremaud and J. H. van Schuppen, *Discrete Time Stochastic Systems: Part I: Stochastic Calculus and Representations,* Report No. 7603a, Department of Systems Science and Mathematics, Washington University, St. Louis (1976).

[8] P. M. Bremaud and J. H. van Schuppen, *Discrete Time Processes: Part II: Estimation Theory,* Report No. 7603b, Department of Systems Science and Mathematics, Washington University, St. Louis (1976).

COHERENT-DETECTION OPTICAL COMMUNICATION SYSTEMS: A PROOF OF THE GAUSSIAN BIT ERROR RATE

SERGIO VERDU
Department of Electrical Engineering
Princeton University
Princeton, New Jersey 08544

Abstract. *It is shown that the asymptotic probability of error of a binary equiprobable hypothesis test for observed Poisson point-processes with rate* $\lambda_i(t) = b_i(t) + (\rho_i(t) + z)^2$, $i = 0, 1$, $z \to \infty$, *is equal to the error probability of optimum deterministic - signal detection in additive white Gaussian noise when the signals coincide with the square root of the point-process rates. This result proves the folk theorem that coherent-detection optical communication systems have Gaussian bit error rates.*

The minimum probability of error of equiprobable binary signal-detection in additive white Gaussian noise is equal to $Q(d/2\sigma)$, where d^2 denotes the energy of the difference between both signals and σ^2 is the noise power spectral density. In contrast, there exists no scalar parameter that characterizes the performance of hypothesis testing for observed Poisson point-processes, and the rates must be known for all times in order to determine error probability. Moreover, unless the log likelihood ratio is conditionally Poisson (e.g. if one rate dominates the other and their ratio is piecewise constant), no closed or semi-closed expressions are known for the probability error.

Fortunately, the cutoff rate, R_0, of a Poisson point-process channel exhibits a much more manageable behavior. Snyder and Rhodes [1] have shown that R_0 coincides with the cutoff rate of a $\sigma = \frac{1}{2}$ white Gaussian noise channel whose signals are equal to the square root of the point-process rates. The purpose of this paper is to show that the error probabilities of both channels also coincide in problems where the point-process rates are uniformly large. Consequently, it is shown that in the limit the error probability depends on the point-process rates only through the energy of the difference between their square roots.

Specifically, it is shown below that the minimum error probability of a binary equiprobable hypothesis test between the point-process rates $\lambda_i(t) = b_i(t) + (\rho_i(t) + z)^2$, $t \in [0, T]$, $i = 0, 1$ approaches $Q\left((\int_0^T (\rho_1(t) - \rho_0(t))^2 dt)^{1/2} \right)$ as

---

This work was partially supported by the U.S. Office of Naval Research under contract N00014-81- K-0146.

$z \to \infty$. This asymptotic rate model is motivated by two important situations arising in optical digital communication systems:

i) In intensity modulated direct-detection systems the rate of the stream of electrons put out by the photodetector is modeled by [2]:

$$\lambda_i(t) = \alpha \, |E_i(t)|^2 + \beta$$

where $E_i(t)$ is the instantaneous complex amplitude of the incident electric field, and α and β are non-negative constants of the photodetector. Identifying $z = \alpha^{1/2} \inf_{t,i} |E_i(t)|$, $\rho_i(t) = \alpha^{1/2}[\,|E_i(t)| - \inf_{t,i} |E_i(t)|\,]$, and $b_i(t) = \beta$, the result of this paper implies that when the minimum value of the envelope of the modulated field is large, the error probability approaches

$$Q\left\{ \left(\alpha \int_0^T (\,|E_1(t)| - |E_0(t)|\,)^2 dt\right)^{1/2} \right\}.$$

ii) In so-called coherent-detection optical communication systems [3], a strong receiver-generated field $|E| \, e^{j\phi(t)}$ is added to the received electromagnetic field $|E_i(t)| \, e^{j\psi_i(t)}$, and the rate of the photodetector output process is

$$\lambda_i(t) = \alpha \left| \, |E| \, e^{j\phi(t)} + |E_i(t)| \, e^{j\psi_i(t)} \right|^2 + \beta,$$

which corresponds to $z = \alpha^{1/2} \, |E|$, $\rho_i(t) = \alpha^{1/2} \, |E_i(t)| \, \cos(\psi_i(t) - \phi(t))$ and $b_i(t) = \beta + \alpha \, |E_i(t)| \, \sin^2(\psi_i(t) - \phi(t))$. Letting $|E| \to \infty$ and applying the result we obtain that the error probability ,conditioned on $\phi(t)$, is given by

$$P_e = Q\left\{ \left[\alpha \int_0^T (\,|E_1(t)| \, \cos(\psi_1(t) - \phi(t)) - |E_0(t)| \, \cos(\psi_0(t) - \phi(t)))^2 dt \right]^{1/2} \right\} \qquad (1)$$

Note that (1) is the minimum probability of error of any receiver that knows $\lambda_1(t)$and$\lambda_0(t)$ (and consequently the phase differences $\psi_1(t) - \phi(t)$and$\psi_0(t) - \phi(t)$).

In the cases where the modulation is either linear $(\psi_1(t) = \psi_0(t) = \psi(t))$ or exponential ($|E_1(t)| = |E_0(t)| = A$), the square of the argument of the Q-function in (1) reduces to $\alpha \int_0^T (\,|E_1(t)| - |E_0(t)|\,)^2 \cos^2(\psi(t) - \phi(t)) \, dt$ and

$\alpha A^2 \int_0^T [1 - \cos(\psi_1(t) - \psi_0(t))] \, [1 - \cos(\psi_1(t) + \psi_0(t) - 2\phi(t))] \, dt$, respectively. (In homodyne PSK with a local coherent source operating at the transmitter frequency, we obtain $Q\left(\sqrt{\frac{A^2 T}{1/4\alpha}} \cos\phi\right)$. A familiar expression to the communications engineer!)

Proposition: Consider the following pair of equiprobable hypotheses:

$H_i : dN_t \,;\, N_t$ is a Poisson counting process with rate $\lambda_i(t, z)$, $t \in [0, T]$, $i = 0, 1$.

Suppose that $\lambda_i(t, z) = b_i(t) + (\rho_i(t) + z)^2$, where $b_i(t) \geq 0$, $\{ | \rho_i(t) | , t \in [0, T]\}$ is bounded, and z is a real number.

Let $d^2 = \int_0^T [\rho_1(t) - \rho_0(t)]^2 \, dt \; (= \lim_{z \to \infty} \int_0^T [\lambda_1^{1/2}(t, z) - \lambda_0^{1/2}(t, z)]^2 \, dt)$. Then, $P_e(z)$, the minimum probability of error of the above test satisfies

$$\lim_{z \to \infty} P_e(z) = Q(d) \tag{2}$$

Proof: Let Ω denote the observation space. For each $z > 0$, let $\{p_i(x, z);\, x \in \Omega\}$, denote the sample function density of a Poisson point-process with rate $\lambda_i(t, z)$, with respect to $\nu()$, the probability measure generated by a unit-rate Poisson point-process, and define the tilted probability density

$$q(x, z) = p_1^{1/2}(x, z) \, p_0^{1/2}(x, z) \exp\big(B(z)\big) \tag{3}$$

where $B(z)$ is the Bhattacharyya distance between $p_1(\cdot, z)$ and $p_0(\cdot, z)$:

$$B(z) = -\ln \int_\Omega p_1^{1/2}(x, z) \, p_0^{1/2}(x, z) \nu(dx). \tag{4}$$

Since $z > 0$ the likelihood ratio $\Lambda(x, z) = p_1(x, z)/p_0(x, z)$ is defined everywhere in Ω, and the minimum probability of error is given by

$$\begin{aligned} P_e(z) &= \frac{1}{2} \int_\Omega \min\{p_1(x, z), p_0(x, z)\} \nu(dx) \\ &= \frac{1}{2} \exp(-B(z)) \int_\Omega \min\{\Lambda^{1/2}(x, z), \Lambda^{-1/2}(x, z)\} \, q(x, z) \nu(dx) \\ &= \exp(-B(z)) \int_\Omega f(\ln \Lambda(x, z)) \, q(x, z) \nu(dx), \end{aligned} \tag{5}$$

where $f(v) = \frac{1}{2} \exp(-|v|/2)$.

We show now that the characteristic function

$$\Phi_z(\omega) = \int_{\Omega} e^{j\omega \ln \Lambda(x, z)} q(x, z) \nu(dx) \tag{6}$$

converges everywhere in R to $exp(-2\omega^2 d^2)$ as $z \to \infty$. Using the fact that the sample function density evaluated at the unordered realization $(t_1, \ldots, t_k)$ is equal to [7]:

$$p_i\left((k, t_1, \ldots, t_k), z\right) = \exp\left[\int_0^T [1-\lambda_i(t, z)]dt\right] \prod_{n=1}^{k} \lambda_i(t_n, z) \tag{7}$$

we can write

$$\Phi_z(\omega) = exp(B(z)) \int_{\Omega} \frac{[p_1(x, z)]^{j\omega + \frac{1}{2}}}{[p_0(x, z)]^{j\omega - \frac{1}{2}}} \nu(dx) =$$

$$= exp(B(z)) \sum_{k=0}^{\infty} \frac{1}{k!} \int_0^T \cdots \int_0^T [exp(-\int_0^T \lambda_1(t, z)dt) \prod_{n=1}^{k} \lambda_1(t_n, z)]^{j\omega + \frac{1}{2}}$$

$$[exp(-\int_0^T \lambda_0(t, z)dt) \prod_{n=1}^{k} \lambda_0(t_n, z)]^{\frac{1}{2} - j\omega} dt_1 \cdots dt_k$$

$$= \exp\left[B(z) - \int_0^T (j\omega + \frac{1}{2}) \lambda_1(t, z) - (j\omega - \frac{1}{2}) \lambda_0(t, z)dt\right] \sum_{k=0}^{\infty} \frac{1}{k!} \left[\int_0^T [\lambda_1(t, z)]^{\frac{1}{2} + j\omega} [\lambda_0(t, z)]^{\frac{1}{2} - j\omega} dt\right]^k$$

$$= \exp\left[\mu(\frac{1}{2} + j\omega, z) - \mu(\frac{1}{2}, z)\right] \tag{8}$$

where we have employed the Chernoff-parameter notation

$$\mu(s, z) = \int_0^T [\lambda_1(t, z)]^s [\lambda_0(t, z)]^{1-s} - s\lambda_1(t, z) - (1-s)\lambda_0(t, z)dt, \quad s \in C. \tag{9}$$

At each time $t \in [0, T]$, the integrand in the exponent of the right-hand side of (8) is equal to

$$[\lambda_1(t, z)]^{\frac{1}{2} + j\omega} [\lambda_0(t, z)]^{\frac{1}{2} - j\omega} - [\lambda_1(t, z)\lambda_0(t, z)]^{1/2} - j\omega[\lambda_1(t, z) - \lambda_0(t, z)]$$

$$= j\omega [2z(\rho_0(t) - \rho_1(t)) + \rho_0^2(t) - \rho_1^2(t) + b_0(t) - b_1(t)] +$$

$$+ [b_1(t) + (\rho_1(t) + z)^2]^{\frac{1}{2} + j\omega} [b_0(t) + (\rho_0(t) + z)^2]^{\frac{1}{2} - j\omega} -$$

$$- [b_1(t) + (\rho_1(t) + z)^2]^{\frac{1}{2}} [b_0(t) + (\rho_0(t) + z)^2]^{\frac{1}{2}}$$

$$= j\omega [2z(\rho_0(t) - \rho_1(t)) + \rho_0^2(t) - \rho_1^2(t) + b_0(t) - b_1(t)] + 2j\omega z(\rho_1(t) - \rho_0(t))$$

$$+ j\omega(\rho_1^2(t) - \rho_0^2(t) + b_1(t) - b_0(t)) - 2\omega^2(\rho_1(t) - \rho_0(t))^2 + o(1)$$

$$= -2\omega^2(\rho_1(t) - \rho_0(t))^2 + o(1). \tag{10}$$

Therefore, on taking the limit of the right-hand side of (8) it follows

$$\lim_{z \to \infty} \Phi_z(\omega) = exp\,(-2\omega^2 d^2). \tag{11}$$

So the log likelihood ratio under $q(x, z)\nu(dx)$ converges in distribution to a zero mean Gaussian random variable with variance equal to $4d^2$. Therefore, the continuity and boundedness of $f(\cdot)$, imply [4] that the limit of the right-hand side of (5) is equal to

$$\begin{aligned}\lim_{z \to \infty} P_e(z) &= [\lim_{z \to \infty} exp\,(\mu(\frac{1}{2}, z))] \int_{-\infty}^{\infty} \frac{1}{4d\sqrt{2\pi}}\, exp\,(-v^2/8d^2 - \mid v \mid /2)dv \\ &= \int_0^{\infty} \frac{1}{2d\sqrt{2\pi}}\, exp\,(-v^2/8d^2 - v/2 - d^2/2)dv \;=\; Q(d).\end{aligned}$$

▽

It has been shown that the asymptotic minimum probability of error of binary hypothesis testing for Poisson point-process observations coincides with the error probability of optimum discrimination of the square root of the point-process rates imbedded in additive white Gaussian noise. The main significance of this result is that the error-rate analysis of optical digital communication systems based on heterodyne or homodyne coherent-detection (which appear to be increasingly important in applications [5],[10]) is equivalent to the analysis of digital signaling over additive white Gaussian noise channels. Interestingly, this result appears to be widely accepted by practitioners in the field on grounds of analytical convenience and agreement with experimental data (e.g., [8],[9]). Attempts to dispatch this result on grounds that "Poisson is Gaussian in the limit" are doomed to failure: the observed point-process is Poisson but is not a Gaussian random process in the limit; and the loglikelihood ratio, although as shown here asymptotically Gaussian under the adequate probability measure, is not Poisson. Also, the fact that high-density shot noise (superposition of replicas of an arbitrary waveform delayed by the point-process arrival times) is a Gaussian process [11] appears to be of little use since no closed-form expression is known for the error probability of a test for mean and autocorrelation.

REFERENCES

[1] D.L. Snyder and I.B. Rhodes, "Some implications of the Cutoff-Rate criterion for coded Direct-detection Optical Communication Systems," *IEEE Trans. Information Theory,* vol. IT-26, no. 3, pp. 327-338, May 1980.

[2] D.L. Snyder, *Random Point Processes,* New York: Wiley, 1975.

[3] R.M. Gagliardi and S. Karp, *Optical Communications,* New York: Wiley, 1976.

[4] K.L. Chung, *A Course in Probability Theory,* Second Edition, New York: Academic, 1974.

[5] H.G. Unger, "Trends in Optical Communications," *Proc. 1984 IEEE International Conf. on Communications,* pp. 153-158, May 1984.

[6] A.D. Wyner and H.J. Landau, "Optimum Waveform Signal sets with Amplitude and Energy Constraints," *IEEE Trans. Information Theory,* vol. IT-30, no. 4, pp. 615-622, July 1984.

[7] P. Bremaud, *Point Processes and Queues: Martingale Dynamics,* New York: Springer-Verlag, 1981.

[8] T. Okoshi, K. Emura, K. Kikuchi, R. Th. Kersten, "Computation of Bit-error rate of various Heterodyne and Coherent-type Optical Communication Schemes," *J. Optical Commun.* vol. 2, no. 3, pp. 89-96, 1981.

[9] R.G. Smith and S. D. Personick, "Receiver Design for Optical Fiber Communication Systems," in *Semiconductor Devices for Optical Communication,* Second Edition, H. Kressel Ed., New York: Springer-Verlag, 1982.

[10] E.E. Basch and T.G. Brown, "Introduction to Coherent Optical Fiber Transmission," *IEEE Communications Magazine,* vol. 23, no. 5, pp 23-30, May 1985.

[11] A. Papoulis, *Probability, Random Variables, and Stochastic Processes.* New York: McGraw-Hill, 1965.

PERFORMANCE CHARACTERISTICS OF LASER DIODE COMMUNICATION SYSTEMS WITH APD RECEIVERS AND COLOR-CODED PPM SIGNALING[†]

M. BAYOUMI AND F. DAVIDSON
Department of Electrical Engineering and Computer Science
The Johns Hopkins University
Baltimore, Maryland 21218

SUMMARY

The performance characteristics of a direct detection optical communication system that has a peak power limited optical transmitter (semiconductor laser diode) and a novel signaling format are presented. The signal format consists of the transmission of a group of L binary source digits as a single light pulse at one of N possible colors (optical center frequencies) in one of M possible time slots. This format is a variation of pulse position modulation signaling referred to as color coded pulse position modulation (CCPPM) signaling. The optical transmitter consists of an array of AlGaAs laser diodes, fabricated to emit radiation at non-overlapping frequencies. At the receiver, the color information is frequency demodulated with a diffraction grating which directs the received light pulse to one of N separate avalanche photodetectors (APD), each located at equal optical path lengths from the grating. The "color" part of the signal alphabet is demondulated with a purely passive component of high efficiency which also serves to prevent unwanted background radiation from reaching the APD photosurface. Time slot demodulation is accomplished in the same manner as in ordinary PPM signaling.

CCPPM signaling has several performance advantages relative to the same alphabet size PPM signaling system. One is a hardware implementation advantage since Q-ary CCPPM signaling requires $N(N-1)/2+NM(M-1)/2$ high speed comparators, whereas ordinary PPM signaling $(N=1,\ M=Q)$ requires $Q(Q-1)/2$ high speed comparators. At fixed binary source data rate, CCPPM signaling requires less received peak optical power to achieve a given symbol error probability than PPM signaling even when intensity fluctuations of the laser source, background radiation, and the nonideal properties of APD photodetectors are taken into account. This advantage arises from the larger time slot sizes found in the CCPPM format relative to those of the same alphabet size PPM system operated at the same source data rate, and was obtained at the expense of a larger emission bandwidth of the laser transmitter array. Excessive amounts of background radiation can negate this advantage, however. The third performance advantage is due to the slow increase in power efficiency, as measured by peak received

power per transmitted information bit, with CCPPM alphabet size Q and N chosen to minimize the total number of comparators needed in the receiver. The power efficiency of an ordinary PPM signaling scheme slowly increases with increasing alphabet size.

[†] Work supported by the National Aeronautics and Space Administration.

ERROR EXPONENTS FOR A DIRECT DETECTION OPTICAL CHANNEL*

PIERRE A. HUMBLET
Massachusetts Institute of Technology
Laboratory for Information and Decision Systems
Cambridge, Massachusetts 02139

ABSTRACT

Error exponent bounds for a noise free optical channel using a direct detection receiver are computed for the cases of Pulse Position Modulation (PPM) and for the basic channel with binary output and average intensity constraint. Some consequences are derived regarding the behavior of the interleaving scheme suggested by Massey, the optimality of the PPM format and the significance of the R_0 parameter.

I. INTRODUCTION

There has been much interest recently [1] in direct detection optical communications, particularly for space applications. Under some conditions [2] a noiseless model is appropriate. In this model, time is divided into slots. During the ith slot the transmitter sends an optical pulse and the receiver counts the number of photons received. That number is assumed to be a Poisson random variable with mean λ_i, where λ_i is proportional to the energy of the transmitted signal. Statistically, the input to this channel is thus a real number (λ_i), the output is an integer (the number of photons) and the transition probability is Poisson, with parameter λ_i. Under an average energy constraint $(E[\lambda_i] \leq \lambda)$ the capacity, in nats/photon units, increases without bound as λ decreases. Throughout the paper we will be interested in the small λ case.

Pierce [1] has suggested that Pulse Position Modulation is effective in that case. In his system, slots are grouped in blocks of M, and a pulse is transmitted during exactly one of the M slots, depending on the information to be transmitted. The receiver can only make an error if no photon is received, which occurs with probability $\varepsilon = \exp(-M\lambda)$. Thus at this "higher level" the channel behaves as a M-ary symmetric erasure channel (SEC) the capacity of which is easily bound to be

$\frac{\ln (M)}{M \lambda}(1-\varepsilon)$ nats/photon. It can be made as large as desired by increasing M while keeping $M\lambda$ constant; unfortunately in doing so the capacity (in nats/slot) decreases. To keep the capacity (in nats/second) constant it is necessary to decrease the slot width, thus increasing the bandwidth and exacerbating the problem of high peak intensity.

McEliece [2] has shown how Reed-Solomon codes can be implemented together with PMM and has analyzed the performance of the joint system. Massey [3] has made the clever observation that if $M = \prod_{i=1}^{m} M_i$ then a M-ary SEC can

---

*Support for the National Science Foundation under grant NSF/ECS 79-20834 is gratefully acknowledged.

be decomposed into m M_i-ary correlated SEC's. He showed that effective coding schemes can be realized by interleaving m codes, especially when the interleaved codes are short constraint length binary convolutional codes.

Most of the theoretical studies of the channels described above have considered the capacity and R_o parameters of the channel. It is striking that R_o is less than one [6] (in nats/photon units) whereas the capacity is unbounded, this has renewed the debate on which parameter best characterizes the practical limitation of a channel.

In this paper we analyze the complete error exponents curves as functions of rate. We start with a brief review of the theory in section II, then we treat the PPM channel (section III) give numerical examples (section IV) and examine the basic binary output channel (section V). This analysis adds insight to Massey's observation, characterizes what parameters of PPM are appropriate for a given λ, and discounts the usefulness of R_o.

II. REVIEW OF ERROR EXPONENTS FOR DISCRETE MEMORYLESS CHANNELS

It is well known [4] that for a given discrete memoryless channel characterized by transition probabilities $P(y|x)$ there exists block codes using the channel N times, transmitting information at the rate of R nats/channel use and meeting the following bounds on the block error probability.

$$P(\text{error}) \leq \exp(-N (E_o(\rho, Q) - \rho R) \qquad 0 \leq \rho \leq 1$$

where

$$E_o(\rho,Q) = - \ln \sum_y [\sum_x Q(x) \; P(y|x)^{\frac{1}{1+\rho}}]^{1+\rho}$$

and $Q(x)$ is a probability distribution on the channel input alphabet. In particular

$$P(\text{error}) \leq \exp(-N E_r(R))$$

where

$$E_r(R) = \max_{0 \leq \rho \leq 1} \max_Q [E_o(\rho,Q) - \rho R]$$

For small ρ, $E_o(\rho,Q) \sim \rho \, I(x;y) + 0(\rho^2)$, where $I(x;y)$ is the mutual information [4] between channel input and output when the input is distributed according to $Q(x)$. Thus the E_r bound gives non trivial result for R up to capacity, where the capacity is defined as $\max_Q I(x;y)$.

Taking $\rho = 1$ yields the weaker "R_o bound":

$$P(\text{error}) \leq \exp(-N(R_o-R))$$

where

$$R_o = \max_Q E_o(1,Q).$$

The R_o bound is trivial when $R>R_o$, which can be much smaller than capacity. However when the optimizing ρ in $E_r(R)$ is equal to 1, which occurs for small R, $E_r(R) = R_o - R$.

These bounds are obtained by analyzing the expected behavior of codes generated randomly, with a per letter distribution Q(x). For small R the E_r bound is not tight as the probability of selecting bad codes dominates the probability of error of good codes. Expurgating bad codes gives the bound [5, problem 3.21].

$$P(\text{error}) \le \exp(-N\, E_x(R))$$

where

$$E_x(R) = \sup_{\rho\ge 1} \rho\left(\max_Q -\ln\left(\sum_x \sum_{x'} Q(x)Q(x') \left[\sum_y \sqrt{P(y|x)P(y|x')}\right]^{1/\rho}\right) - R\right)$$

Moreover, asymptotically over N, the following sphere packing bounds [4] applies to all codes

$$P(e) \gtrsim \exp(-N\, E_{sp}(R))$$

where

$$E_{sp}(R) = \max_{0\le\rho} \max_Q (E_o(\rho,Q) - \rho R)$$

Note that if $E_{sp}(R)$ is achieved by $\rho \le 1$, then $E_{sp}(R) = E_r(R)$. Thus the E_r and E_{sp} bounds are asymptotically tight when the optimal ρ is small, which can be shown to correspond to large R. When R is small it has been conjectured that the maximum of E_r and E_x is also asymptotically tight.

III. APPLICATION TO THE M-ARY SYMMETRIC ERASURE CHANNEL

We specialize the error exponents to the M-ary symmetric erasure channel. In all cases the optimizing input probability assignment is uniform so that we change our notation, removing the argument Q but adding M, thus defining a straightforward fashion $E_o(\rho,M)$, $R_o(M)$, $E_r(R,M)$, $E_x(R,M)$, $E_{sp}(R,M)$. We are very fortunate that the optimization over ρ leads to closed form results for E_{sp} and E_r. In fact:

$$E_o(\rho,M) = -\ln\left(\varepsilon + (1-\varepsilon)M^{-\rho}\right)$$

$$E_{sp}(R,M) = -R'\ln\left(\frac{1-\varepsilon}{R'}\right) - (1-R')\ln\frac{\varepsilon}{1-R'} \qquad 0 \le R' \le 1-\varepsilon \tag{1}$$

where $R' = R/\ln M$.

(Note that E_{sp} is Kullback's information divergence between R' and 1-ε!) This indirectly shows that the channel capacity is

$$C = (1-\varepsilon)\ln M \qquad (2)$$

One also finds

$$R_o(M) = -\ln(\varepsilon + (1-\varepsilon)M^{-1})$$

and

$$E_r(R,M) = \begin{cases} E_{sp}(R,M) & R > \frac{1-\varepsilon}{M\varepsilon + 1-\varepsilon}\ln(M) \\ R_o(M) - R & \text{else} \end{cases}$$

Finally

$$E_x(R,M) = \max_{\rho \geq 1} \left(-\rho \ln (M^{-1} + (1-M^{-1})\varepsilon^{1/\rho}) - \rho R\right)$$

We now turn to Massey's observation that the M-ary SEC can be decomposed into $\frac{1}{p}$ $(0 < p \leq 1)$ M^p-ary completely correlated SEC's. Instead of using a (N,R) code for the large channel one can equivalently use (N,pR) codes for the subchannels and still achieve the same total information rate.

Exponential bounds derived previously still hold for the subchannels, using the proper R and M. We make two observations:

a) $E_{sp}(R,M) = E_{sp}(pR, M^p)$

thus the channel and subchannels have the same sphere packing exponent. However the more subchannels one uses (the smaller p), the smaller is the rate region where E_{sp} is equal to E_r.

b) From (1) $E_o(\rho,M) = E_o(1,M^\rho)$

thus if $E_{sp}(R,M)$ is achieved by $\rho = p$, then

$$E_{sp}(R,M) = E_o(1,M^p) - pR.$$

In other words the curve $E_{sp}(R,M)$ could be generated as the upper envelope of the R_o bound for subchannels, if it were not for the diophantine constraints on 1/p and M^p. This is illustrated in Figure 1 where E_{sp}, E_r, and E_x are displayed. For easy normalization as M varies, we plot them as a function of $R' = R/\ln M$ and display their value in "nats/channel use" divided by $-\ln \varepsilon$, which in the optical channel context is interpreted as "nats/photon".

IV. APPLICATION TO PRACTICAL CODES

The error bounds presented in the previous section were generated under random coding assumptions. They are not expected to accurately predict the performances of good codes. However in the following examples E_r turns out to be much more useful than R_o when it comes to comparing codes.

For example Massey [3] noted that a 16-ary N=15 K=8 Reed-Solomon code performs as well as a M=16 channel as 4 interleaved N=24 K=12 binary Golay codes with maximum likelihood decoding (K denotes the number of information digits in a codeword). To maintain performance while going from M=16 to M=2 it was thus necessary to increase N from 15 to 24, as qualitatively predicted by the theory in the previous section. We now get to verify this quantitatively.

In the table below we compare the Reed-Solomon and Golay systems for rates R = 2/3 nats/photon and .5 nats/photon. The relationship between ε and R is given by $-\ln \varepsilon = \frac{K}{N}\,\frac{\ln 16}{R}$. The actual probabilities of error for the codes at hand were computed according to the methods outlined in [2] and [3]. We obtain the following values

R	System	ε	R_o	$\mathrm{Max}(E_r, E_x)$	bound on P(e)	Actual P(e)
2/3	Reed-Solomon	.109	1.80	.406	$2.3\ 10^{-3}$	$5.3\ 10^{-5}$
	Golay	.125	.58	.229	$4.1\ 10^{-3}$	$4.5\ 10^{-5}$
.5	Reed-Solomon	.052	2.20	.718	$2.1\ 10^{-5}$	$2.3\ 10^{-7}$
	Golay	.062	.63	.306	$6.48\ 10^{-4}$	$1.8\ 10^{-7}$

As expected, due to the small value of N, the bounds are not very good. However they correctly indicate the equivalence of the first two systems. The agreement is not as good between the last two but these cases are very much in the low rate region where E_r and E_x are too small. R_o also appears in the table above but it does not explain the phenomenon we have just examined.

V. THE BINARY OUTPUT POISSON CHANNEL

Instead of restricting ourselves to PPM we now take a more fundamental look at the problem. We relax the PPM format and allow independent pulses in every slot. The receiver is still binary, i.e. all decisions are based on whether any photon is received in a slot. We show that the pulse energy distribution optimizing E_o has mass only at 0 and another point. We also show that at high rates (bits/photon) there is almost no loss in using PPM.

The average intensity per interval must be constrained for the problem to make sense and this complicates the computation of the error exponents. An upperbound on the probability of error can still be obtained by maximizing $E_o(\rho,Q)$ over all input distributions satisfying the constraint. The resulting bound will be non trivial for rates up to capacity, but it will not be very good. Gallager [4, section 7.3] explains how a better bound (the true $E_r(R)$) can be obtained by introducing and optimizing an extra parameter. We limit ourselves to the simple bound. This is usually also how R_o is defined in presence of constraints.

We first show that the distribution maximizing $E_o(\rho,Q)$ has mass only at 0 and at another point. As this holds for all $\rho > 0$ it is also true for the distribution achieving capacity.

We wish to minimize

$$\exp(-E_o(\rho,Q)) = (\int_0^\infty dQ(x)e^{-x/1+\rho})^{1+\rho} + (\int_0^\infty dQ(x)(1-e^{-x})^{\frac{1}{1+\rho}})^{1+\rho}$$

subject to the constraints

$$\int_0^\infty x\, dQ(x) \leq \lambda$$

$$Q(\infty) = 1$$

$$Q(x) \text{ non decreasing}$$

As the right hand side above is convex in Q, we can use the Kuhn-Tucker necessary and sufficient conditions for optimality:

$$(\int dQ(y)e^{-y/1+\rho})^{\rho}\, e^{-x/1+\rho} + (\int dQ(y)(1-e^{-y})^{\frac{1}{1+\rho}})^{\rho}(1-e^{-x})^{\frac{1}{1+\rho}} \tag{3}$$

$$\geq -ax + b \qquad \forall\, x \geq 0$$

for some $a > 0$ and b, with equality with probability one w.r.t. Q(x). For arbitrary non negative q_o, q_1 and ρ the function $q_o\, e^{-\frac{x}{1+\rho}} + q_1(1-e^{-x})^{\frac{1}{1+\rho}}$ has a second derivative with respect to x equal to

$$q_o[(\frac{1}{1+\rho})^2\, e^{-\frac{x}{1+\rho}}] - q_1[\frac{\rho}{(1+\rho)^2}\, (1-e^{-x})^{-(\frac{1+2\rho}{1+\rho}}\, e^{-2x} + \frac{1}{1+\rho}\, (1-e^{-x})^{-\frac{\rho}{1+\rho}}\, e^{-x}]$$

This last function has a single zero in $(0,\infty)$, as the term in the first bracket decreases more slowly than the ones in the second. Thus the left hand side of (3) cannot touch the right hand side in more than one point in $(0,\infty)$. This shows that the optimizing Q has a mass, say $1-p$, at $x = 0$, and mass p at λ/p.

We now turn to the channel capacity (per channel use). It can be written as

$$C = \max_{0\le p\le 1} (H(y) - H(y|x))$$

$$= \max_{0\le p\le 1} H(p(1-e^{-\lambda/p})) - p_1 H(1-e^{-\lambda/p}) \qquad (4)$$

where H denotes the binary entropy function. Choosing $p = \lambda$ yields

$$C \ge H(\lambda(1-e^{-1})) - \lambda\, H(1-e^{-1})$$

The non negativity of H yields the upperbound

$$C \le \max_{0\le p\le 1} H(p(1-e^{-\lambda/p})) = \begin{cases} \ln 2 & 1-e^{-\lambda} > .5 \\ H(1-e^{-\lambda}) & \text{else} \end{cases}$$

An approximation can be obtained by keeping the low order terms in the power series expansion of the right hand side of (4):

$$C = \max_{0\le p\le 1} \lambda \left[\frac{\lambda}{2p} - \ln p + \frac{\lambda}{2}\,\frac{\ln p}{p} + 0(\lambda)\right] \text{ nats}$$

The optimizing p satisfies the relation

$$p^* = \frac{\lambda}{2} \ln \frac{1}{p^*}$$

thus

$$p^* = \frac{\lambda}{2}\left[\ln\frac{1}{\lambda} - \ln\ln\frac{1}{\lambda} + 0\left(\frac{\ln\ln\frac{1}{\lambda}}{\ln\frac{1}{\lambda}}\right)\right]$$

and

$$C = \lambda\left[\ln\frac{1}{\lambda} - \ln\ln\frac{1}{\lambda} + 0\left(\frac{\ln\ln\frac{1}{\lambda}}{\ln\frac{1}{\lambda}}\right)\right] \text{ nats}$$

Comparing this with McEliece's bound [2, (4.5)]

$$C \le \lambda\left[\ln\left(\frac{1}{\lambda} + 1\right) + 1\right]$$

for multilevel channel output shows that, as expected, restriction to binary output entails no asymptotic loss in capacity as λ decreases.

Finally, when one is given an optical channel with a given λ per interval and one wishes to use Pierce's PPM scheme, the question arises on how large to make M. The optimal values of p found above suggests that M in the range $\dfrac{2}{\lambda \ln \frac{1}{\lambda}} \le M \le \dfrac{1}{\lambda}$ would be suitable. Figure 2 shows the error exponent E_r of various PPM channels, when $\lambda = 10^{-3}$ per slot, and compares them with the bound on E_r derived in this section. The true E_r would always dominate, but its bound does not, especially at low rates. $M = 361$ is the value of M maximizing the capacity per photon when $\lambda = 10^{-3}$ as can be found by going back to (2). One concludes that a well chosen PPM format is close to optimality not only in terms of capacity, but also as far as error exponents are concerned.

In the intermediate rate region ($R \sim \frac{C}{2}$), $M \simeq 1/\lambda$ does about as well as the M maximizing the capacity of the channel. The advantage of $M \simeq 1/\lambda$ is that $\varepsilon \simeq e^{-1}$, thus one expects usual codes, such as rate 1/2 Reed-Solomon [3], to perform well. Being closer to capacity requires a smaller M, thus a larger and low rate codes tolerating channels that erase most symbols.

For example, when $M = \frac{1}{\lambda}$, $E_{sp} = E_r = .205$ nats/photon (or per PPM channel use) for $R = .5(1-e^{-1})\ln\frac{1}{\lambda}$ nats/photon (or per PPM channel use); this rate is half the capacity of the PPM channel, or about $.5(1-e^{-1})$ of the capacity of the binary output photon channel.

A probability of error of $2\ 10^{-6}$ can be achieved by using $N = 64$. The theory of section III predicts that 4-ary codewords of that length (with $K \simeq 20$) can be interleaved on the $M = \frac{1}{\lambda}$ PPM channel while still achieving the same probability of error.

VI. THE SIGNIFICANCE OF R_o

As indicated previously R_o is the maximum value of $E_r(R)$, $R_o = E_r(0)$ or possibly less when there are constraints on the input signal. The notation is unfortuate as "R_o" connotes the idea of "rate", not of value of error exponent. This is probably due to the fact that the "R_o bound" is non trivial only for $R < R_o$, but this has little significance as the R_o bound is not tight.

For the Poisson channel any lower bound on error exponents cannot be larger than 1/photon (when base e is used; we hesitate writing "nats/photon"): as any code with P equally likely codewords and an average of λ photons/codeword has an average error probability of at least $(1-\frac{1}{P})\exp(-\lambda(\frac{P}{P-1}))$ due to the possibility of confusion when no photon is detected. Thus it is not surprising that R_o be less than 1 nat/photon. R_o approaches 1 in the limit of few nats/dimension, when the erasure probability dominates the probability that randomly selected codewords are "close".

The boundedness of error exponents has no implications on the value of the capacity, which has been shown to be arbitrarily large on a bit per photon basis. It has no implication on decoding complexity either: the analysis reported above shows that there is essentially no loss in implementing Massey's interleaving scheme. Increasing M while keeping ε fixed and interleaving log M fixed (N,R) codes allows arbitrarily high rates per photon while keeping the probability of error and the amount of decoding computation per bit constant.

Finally it has been suggested [3] that R_o be considered as a rough measure of the necessary code length in channel symbols required to achieve a given probability of error. The example reported in section IV tends to show that this role belongs more properly to $E_r(R)$ and $E_x(R)$ whose values bear little relationship to that of R_o.

We thus conclude that in general R_o has little significance for communication systems, except as it relates to sequential decoding of convolutional codes [5]. Also, the practical limit on the achievable rate per photon on the optical channel lies not in the necessary complexity of the decoders, or in other information theoretic concepts, but on purely physical reasons: the difficulty to generate the vaninshingly narrow but powerful pulses required [1], [7].

ACKNOWLEDGMENT

This paper benefits from many suggestions made by reviewers of a previous version.

REFERENCES

[1] J. R. Pierce, "Optical Channels: Practical Limits With Photon Counting", IEEE Trans. Commun., Vol. COM-26, pp. 1819-1821, Dec. 1978.

[2] R. J. McEliece, "Practical Codes for Photon Communication", IEEE Trans. Inform. Theory, Vol. IT-27, No. 4, July 1981.

[3] J. L. Massey, "Capacity, Cutoff Rate, and Coding for a Direct-Detection Optical Channel", IEEE Trans. Commun., Vol. COM-29, No. 11, pp. 1615-1621, Nov. 1981.

[4] R. G. Gallager, Information Theory and Reliable Communication, Wiley and Sons, New York, 1968.

[5] A. J. Viterbi and J. K. Omura, Principles of Digital Communication and Coding, New York, McGraw-Hill, 1978.

[6] D. L. Snyder and I. B. Rhodes, "Some Implications of the Cutoff-Rate Criterion for Coded Direct-Detection Optical Communication Systems", IEEE Trans. Information Theory, Vol. IT-26, No. 3, May 1980.

[7] S. A. Butman, J. Katz, and J. R. Lesh, "Practical Limitations on Noiseless Optical Channel Capacity", Jet Propulsion Lab., Pasadena, CA, DSN Progress Rep., Vol. 42-55, pp. 12-14, Nov.-Dec., 1979.

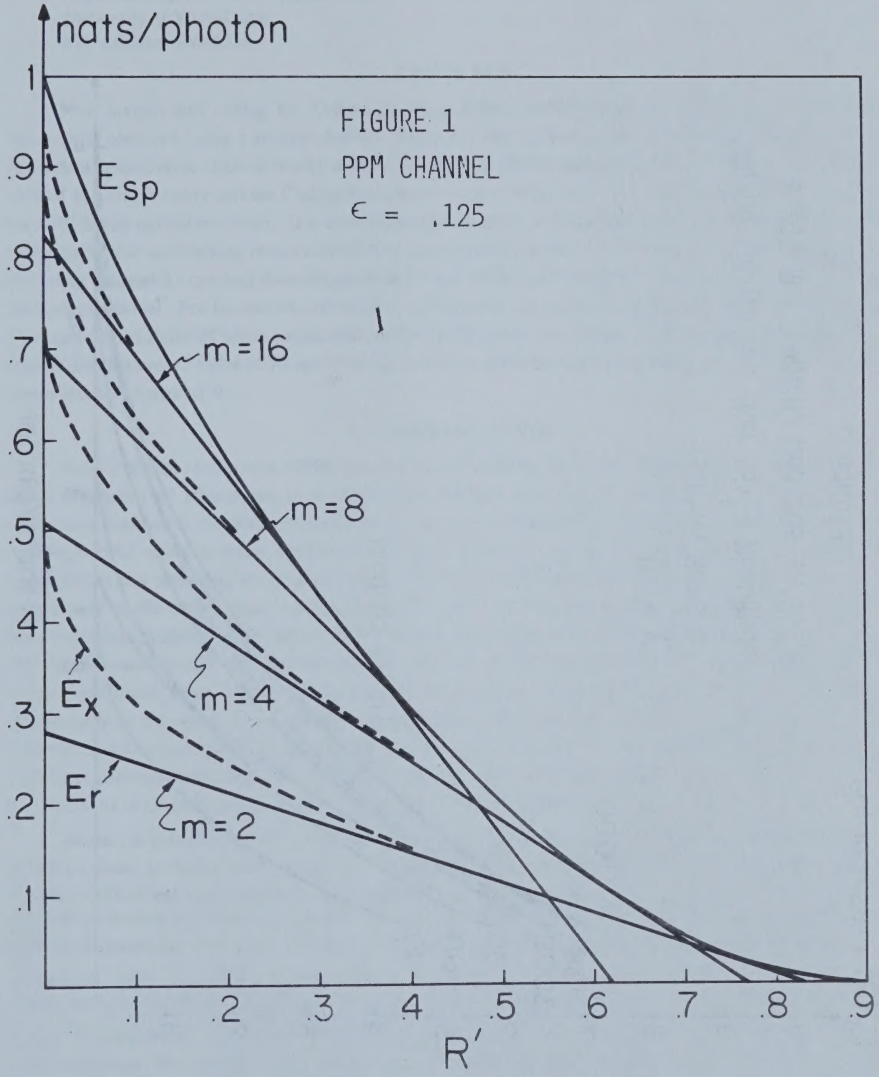


FIGURE 1
PPM CHANNEL
$\epsilon = .125$

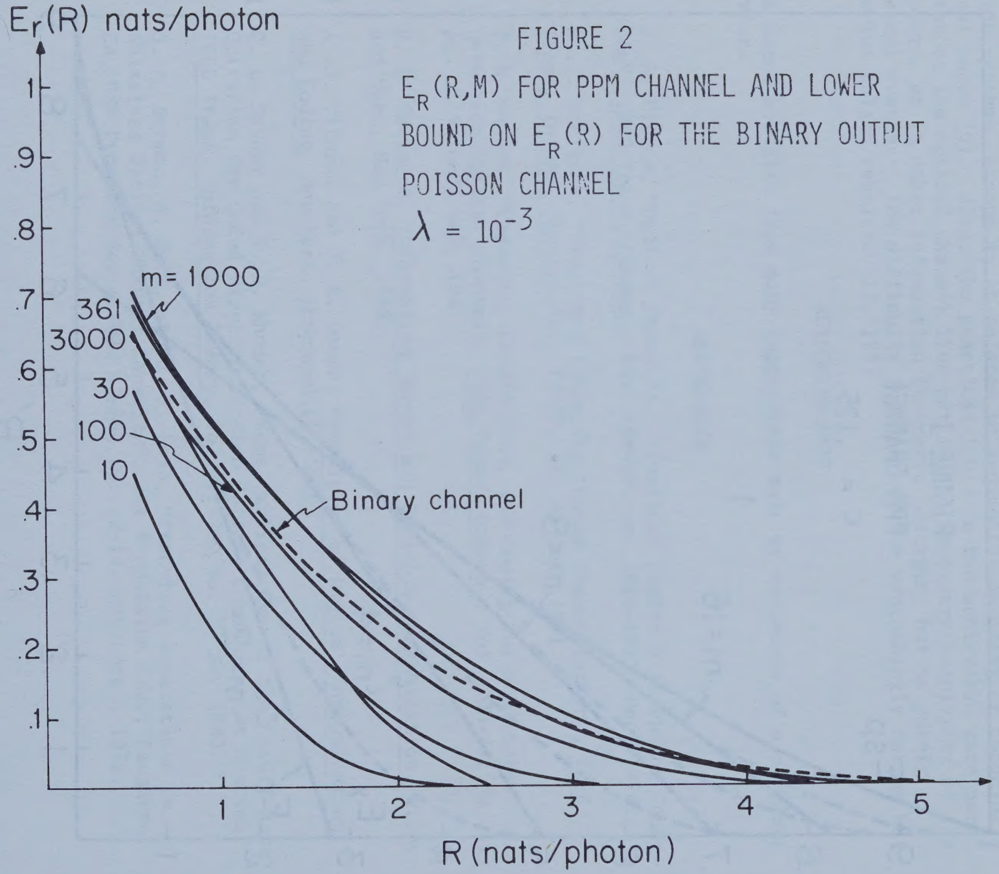


FIGURE 2

$E_R(R,M)$ FOR PPM CHANNEL AND LOWER
BOUND ON $E_R(R)$ FOR THE BINARY OUTPUT
POISSON CHANNEL
$\lambda = 10^{-3}$

MULTI-PULSE PPM AND A NEW LOOK AT CODING FOR DIRECT-DETECTION OPTICAL CHANNELS USING APD BASED RECEIVERS

MARK A. HERRO and LAIZHAO HU
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

ABSTRACT

New insight into coding for Avalanche Photo Diode (APD) based Direct-Detection Optical (DDO) channels is obtained using a simple channel model that clearly points out the difference between the signal dependent optical noise channel model and the well known AWGN channel model. Coding and modulation are viewed as a single entity and the Coding/Modulation Gain (CMG) is used to analyze and construct good codes for APD based optical receivers. It is shown that the structure of block and convolutional codes renders them fairly inefficient at obtaining reasonable CMG for the optical channel. In contrast, codes (or modulations) that use fewer channel 1's (pulses) than information 1's per block (such as PPM) can offer substantial CMG's on the optical channel. For bandwidth constrained applications, we propose using PPM with more than one pulse per word (Multi-Pulse PPM) to obtain reasonable CMG while consuming small amounts of bandwidth. The comparison between 2 Pulse-PPM and PPM shows that a 3 dB decrease in coding gain can reduce the required bandwidth by a factor of 10.

I. INTRODUCTION

Pulse Position Modulation (PPM) has been used for direct-detection optical communications, because it offers a number of advantages over other types of light wave communication. It is relatively easy to implement, maximizes the channel cutoff rate [1], and is easily matched to Reed-Solomon (RS) outer codes [2]. Although PPM offers superior performance, it does so by consuming large amounts of bandwidth. When bandwidth is at a premium, we propose using 2 Pulse PPM (2P-PPM), which contains 2 pulses in a codeword as opposed to the single pulse used with PPM. The line between coding and modulation in a digital communication system is not at all clear [3]. In fact, we can view the PPM as a code in the sense that it buys performance with bandwidth. As Massey [3] suggests, we refer to the gain of the coding and modulation system combined, since only when the two are designed in a coordinated way can we expect maximum efficiency from the system. In order to determine which codes perform well on the optical channel, we apply different codes to both the DDO channel and the AWGN channel and compare their Coding/Modulation Gain (CMG) at different error rates. We find that the two channels yield significantly different results. This insight has guided us in choosing codes that are particularly well matched to the DDO channel.

In section two, we present a simple model for the DDO channel to facilitate the coding analysis. Then in section three, we define the Channel Bit Voltage Ratio (CBVR), which will be used in studying the CMG of both the AWGN and DDO channels. In section four, we further structure the concept of CMG by introducing the code pulse-rate, R_p, which is defined as the ratio of the number of 1's per information word to the number of 1's transmitted per codeword. We show that the CMG is proportional to R_p and hence when bandwidth is unlimited, PPM provides the best choice for both the AWGN and DDO channels, since of all coding/modulation systems, PPM maximizes R_p. PPM's large bandwidth consumption motivated us to search for more "economical" coding/modulation systems. In section five, we propose using more than one pulse per PPM codeword. We call this code Multi-Pulse PPM and compare its performance with that of PPM and conventional block and convolutional codes over both the AWGN and DDO channels. In situations where bandwidth is limited, Multi-Pulse PPM appears to be an attractive alternative to PPM or conventional error-control codes.

This work was supported by NASA Grant NCA2-1R562.

II. THE AWGN CHANNEL AND THE DDO CHANNEL MODELS

When a transmission is based on On-Off Keying (OOK), the resultant channel formed by the physical channel and the modulation system has only two inputs- either a **1** (the signal is on) or a **0** (the signal is off), and a continous output. If we further adopt hard decisions at the receiver, the channel has only two outputs- a **1** or a **0**. In such a case, the channel can be modeled as the *binary non-symmetric channel* shown in Figure 1(b). Looking at the unquantized receiver output voltage, we observe that the voltage will have a certain distribution under the condition that either a **1** or a **0** is sent, as shown in Figure 1(a). The threshold setting T is used to make the decision as to what was sent. When the output voltage is greater than T, the decision is that a **1** is sent. Otherwise, the decision is that a **0** is sent. Integrating the density curve p(y|x=**0**) from the threshold T to ∞ will give the error probability δ, of decoding **1** given **0** was sent. Likewise, the area under the density curve p(y|x=**1**) from -∞ to T will give the error probability ε, of decoding **0** when **1** was sent. These error probabilities for the physical channel, along with their significance in the mathematical model are illustrated in Figure 1.

For different physical channels, and for different modulation schemes, the output voltage distribution will be different. For the well known AWGN channel, both of the density functions (under the condition that a **1** is sent and that a **0** is sent) are Gaussian, and they have the same variance. After normalization, we can always take the variance to be 1, the mean when a **0** is sent to be 0, and the mean when a **1** is sent to be some normalized signal strength S. We distinguish the parameters of the AWGN channel with the subscript "a". Note that since S_a is a voltage, it is proportional to the square root of the signal energy used for sending a **1**. Under these conditions, we can write

$$S_a = b_a(E)^{1/2} \tag{1}$$

where b_a is a constant of proportionality for the AWGN channel,

$$\text{and} \qquad \delta = Q(T), \qquad \varepsilon = Q(S_a - T) \tag{2}$$

$$\text{where} \qquad Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{x^2}{2}} dx \tag{3}$$

However, for the APD based DDO channels, the situation is quite different. The noise is neither Gaussian, nor independent of the signal. When we are only considering the error performance of the channel,the Gaussian density forms a reasonable approximation to the density functions of the receiver output voltage ([5], [6], [7], [8] and [9]). This will yield a channel with two states, indexed by whether a **0** or a **1** was sent. In both states, the receiver output voltage has a Gaussian distribution. Their means and the variances are given in [5].

$$\text{0 state:} \begin{cases} \overline{V_{out}} = K_1 (\frac{E_0}{hf} M) & (4) \\ \sigma^2 = K_1^2 (\frac{E_0}{hf} M^2 F + Z^2) & (5) \end{cases}$$

$$\text{1 state:} \begin{cases} \overline{V_{out}} = K_1 (\frac{E_r}{hf} M) & (6) \\ \sigma^2 = K_1^2 (\frac{E_r}{hf} M^2 F + Z^2) & (7) \end{cases}$$

where hf is the energy of a photon, E_0 and E_r are the received energies in state **0** and state **1** respectively, M is the mean avalanche gain, F is the avalanche detector excess noise factor, and Z is the preamplifier noise parameter, which varies according to the quality of the receiver preamplifier. The excess noise factor, F, is reasonably approximated by

$$F = k \cdot M + (2 - 1/M)(1 - k) \tag{8}$$

where *k* is the ionization ratio of the avalanche diode.

We can normalize this model by letting the mean when a **0** is sent be 0, and the variance when a **0** is sent be 1 ($\sigma_0^2=1$). Then the mean when a **1** is sent will be equal to the normalized signal strength S, and we denote the variance when a **1** is sent by σ_1^2, where

$$\sigma_1^2 = 1 + F'S \tag{9}$$

$$F' = F/c \tag{10}$$

$$S = (E_r - E_0)/hf/c \tag{11}$$

$$c^2 = (E_0/hf) \cdot F + (Z/M)^2 \tag{12}$$

Note that F' is a parameter that indicates the non-symmetricity or non-additivity of the channel, i.e., how dependent the noise is on the signal. If F'=0, the noise when a **1** is sent is the same as that when a **0** is sent. That implies that the noise is independent of the signal. When F'≠0, the noise is signal dependent. Calculated from some typical parameters([5], [9]), we can see that values of F' on a typical DDO channel range from 0.1 to 2.

There are two major differences between the DDO channel and the AWGN channel. The first is that the normalized signal strength, S, is not proportional to the square root of the signal energy as in the AWGN channel, but directly proportional to the signal energy. This can be seen from (11), and we denote this by

$$S_d = b_d \cdot E_d \tag{13}$$

where the subscript "d" implies the channel is a DDO channel, and $E_d = E_r - E_0$ is the effective signal energy. When $E_0=0$, $E_d = E_r$ is exactly the signal strength. The second difference is that the noise on the DDO channel is usually non-additive. We can not always assume that F'=0. So the error in decoding **0** given **1** was sent is

$$\varepsilon = Q[(S_d - T)/\sigma_1] \tag{14}$$

where σ_1 depends on S_d.

Figure 2 pictorally illustrates these two differences in the channel models. In section four we will see that the difference between these two channels becomes significant when coding is applied.

III. INVARIANT CODES AND THE CBVR OF A CODE

Recall the definition of the fractional weight of binary sequences.

Definition 1. For a binary sequence, define the fractional weight (W) of a sequence as the percentage of **1**'s in the sequence.

Definition 2. An **invariant code** is a mapping from the set of information sequences to the set of code sequences, that preserves the fractional weight of the set of uncoded information sequences, provided the fractional weight of the uncoded sequence is 0.5.

Theorem 1. A linear code is an invariant code iff its K·N generator matrix has no all **0** columns.

Proof: Consider the first channel bit of a codeword. Without loss of generality, if there are no all **0** columns, we can assume that the first row of the generator matrix has a **1** in the first column. The set of all information sequences beginning with a **0** will result in a set of codewords whose first bit will be a **1** with some probability, A. Then for the set of codewords generated by information sequences starting with a **1**, the probability of the first bit being a **1** will be 1-A, since the code is linear. When the total fractional weight of all of the uncoded (information) sequences is 0.5, every codeword will appear with equal probability. This means that the number of codewords beginning with a **1** is equal to the number of codewords beginning with a **0**, and so the probability of being in one set is the same as being in the other. As a consequence, the fractional weight of the first codeword bit is 0.5·A+0.5·(1-A) = 0.5. The same argument holds for every channel bit when the generator matrix has no all **0** columns. Therefore, the total fractional weight of the codewords is 0.5 and equal to the total fractional weight of the information sequences. It then follows that the code is invariant if its generator matrix has no all **0** columns. On the other hand, if the generator matrix of the code has at least one all **0** column, the channel bit corresponding to that column will always be **0**. This will lower the total fractional weight of the code sequences. As a consequence, the total fractional weight of the coded sequences is strictly less than 0.5, which means that the code is not invariant.

Theorem 2. All (N,1,M) convolutional codes that use the current information bit to generate each channel bit are invariant.

Proof: Regardless of what is in the encoder's memory, when the current information bit is a **1**, the N output bits of the encoder will be the complement of the output produced when the information bit is a **0**. When the source generates **1**'s and **0**'s with equal probability, we know (by an argument similar to that in the proof of Theorem 1) that the total fractional weight of the encoded sequences will be 0.5. Since the total fractional weight of the uncoded sequences is also 0.5, the code is invariant.

Most of the linear codes are invariant codes. On the other hand, non-linear codes are more likely not to be invariant. For example, L-ary Pulse Position Modulation (PPM), when viewed as a code, has a codeword fractional weight of 2^{-L}, and hence is invariant only when L=1, if we assume the source produces **1**'s and **0**'s with equal probability.

Definition 3. For an OOK based binary channel, and a target bit error rate p_b, define the ratio of the signal strength (in voltage) required for a channel **1** (pulse) when no coding is used to that when coding is used, as the **Channel Bit Voltage Ratio (CBVR)** of the code for that error rate, i.e.,

$$\text{CBVR} = S_u/S_c \tag{15}$$

where S_u and S_c are the signal strengths (voltages) for the uncoded and the coded systems respectively.

Example 1. The CBVR of a (7,4) Hamming code over an AWGN channel.

We assume that the fractional weight of the uncoded sequence is 0.5, which is the most common case. (We only discuss such cases in this paper.) By Theorem 1, we know that the fractional weight of the coded sequences is also 0.5. Under this assumption, the best decision threshold is T = S/2 regardless of whether coding is used or not. By (2) and (3), we have

$$\delta = \varepsilon = Q(S/2) \tag{16}$$

Denote the target bit error rate as p_b. Without coding, it is obvious that the required signal strength is

$$S_u = 2Q^{-1}(p_b), \tag{17}$$

since $p_b=\varepsilon=\delta$. With coding, if we denote the original channel crossover probability by ε, we have

$$S_c = 2Q^{-1}(\varepsilon) \tag{18}$$

As the Hamming code is single error correcting, the resulting block error rate is upper bounded [10] by

$$p_B \le 1- (1-\varepsilon)^7 - 7\varepsilon\cdot(1-\varepsilon)^6 \tag{19}$$

For a non-systemetic code, approximately half of the bits in a bad block are incorrect, and the bit error rate is upper bounded by

$$p_b \le 0.5\cdot p_B \le 10.5\varepsilon^2 - (35\varepsilon^3-52.5\varepsilon^4) - (42\varepsilon^5-17.5\varepsilon^6) -3\varepsilon^7 \tag{20}$$

Since the last three terms are all positive, we have

$$p_b < 10.5\,\varepsilon^2 \tag{21}$$

As long as $p_b < 1/10.5$ (or $\varepsilon < 1/10.5$), it is guaranteed that $p_b < \varepsilon$. As it is well known that Q is a monotonically decreasing function, we can see that the signal strength needed with coding is smaller than that without the code, or that the CBVR is greater than 1 at least when $p_b < 1/10.5$ (or $\varepsilon < 1/10.5$).

Example 2. The CBVR of a (2,1,2) convolutional code over an AWGN channel.

We assume that the fractional weight of the uncoded sequence is 0.5 as before. By Theorem 2, we know that the fractional weight of the coded sequence is also 0.5. So (17) and (18) are still valid, since T=S/2. Using the transfer function bound [11], we have

$$p_b < \left.\frac{x^5}{(1-x)^2}\right|_{x=2\sqrt{\varepsilon(1-\varepsilon)}} \tag{22}$$

It is not difficult to observe that $p_b < \varepsilon$ when $p_b < 0.038$ (or $\varepsilon < 0.038$). Using the same argument as before, we see that the CBVR of this code is greater than 1 at least when $p_b < 0.038$ (or $\varepsilon < 0.038$).

The above two examples site situations where the CBVR is greater than 1. This results, not from the fact that the channel is AWGN, but from the structure of the codes. As a matter of fact, the conventional block codes and convolutional codes all have CBVR greater than 1, when p_b is small. This fact will be used later in section four.

Example 3. The CBVR of a 2-ary PPM over an AWGN channel.

PPM is a form of modulation, but it can also be viewed as a code. For L-ary PPM, we encode L information bits by sending one **1** among the 2^L bits (time slots) of a word. For 2-ary PPM, there are four information words: 00, 01, 10, 11. We encode them into four codewords: 0001, 0010, 0100, 1000. We still assume that the uncoded sequences have a fractional weight of 0.5, so (17) is valid. But now the fractional weight of the coded sequences is no longer 0.5, but is equal to 0.25. It should be pointed out that the best threshold here is no longer T =S/2, as can be expected [12]. For different decoding rules for PPM, the best threshold will vary. With "errors only" decoding, we obtain [12] a block error probability of

$$p_B = 1 - 0.25\varepsilon(1-\delta)^3 - (1-\varepsilon)\,[(1-\delta)^3 + 1.5\delta(1-\delta)^2 + \delta^2(1-\delta) + 0.25\delta^3\,] \tag{23}$$

We numerically calculated the threshold that minimized the error rate at any given signal strength. If we assume that in the wrong block, half of the bits will be wrong, we can estimate the bit error probability by $p_b=0.5\cdot p_B$. For a given target p_b, using (2) and (23) we can find the required signal strength. Some numerical examples are shown in Table 1. From these we can see that for PPM, the CBVR is less than 1, which is different from the result for conventional codes. More general principles of decoding PPM has been studied [7], and it turns out that neither the size of the PPM word, nor the differences in the decoding rule significantly effect the CBVR.

IV. CODING/MODULATION GAIN ANALYSIS

Since it is difficult to distinguish coding from modulation, we define the combined Coding/Modulation Gain as follows.

Definition 4. Let E_u be the energy per information bit required to reach an error rate p_b in an uncoded system, and E_c be the energy per information bit required to transmit the sequence with the same reliability using a code and its associated modulator. We define the **Coding/Modulation Gain (CMG)** as the ratio of E_u/E_c and usually express it in dB, i.e.,

$$\mathrm{CMG} = 10\cdot\log_{10}(E_u/E_c) \tag{24}$$

The CMG is a function of the coding/modulation scheme and the required error rate. In addition, we will see that it is also a function of the channel. For OOK based channels, we introduce the following code-pulse rate, R_p, to facilitate the analysis of the CMG.

Definition 5. Define the ratio of the number of **1**'s in a sequence before coding to that after coding as the **code-pulse rate**, R_p.

If we use K and N as the total number of bits before and after coding respectively, and W_u and W_c as the fractional weight of the sequences before and after coding respectively, then the dimensionless code rate is given by

$$R = K/N \tag{25}$$

while the code-pulse rate is written as

$$R_p = (K\cdot W_u)/(N\cdot W_c) = R\,(W_u/W_c) \tag{26}$$

As mentioned earlier, we only discuss the case when the fractional weight of the uncoded information sequence is 0.5, that is W_u=0.5. It is clear that if the code is invariant, we simply have R_p=R, because W_c is also 0.5. However, since not all codes are invariant, the ratio of W_u/W_c can be greater than 1 if W_c<0.5. In other words, some efficient error-control codes have R_p>1. For example, L-ary PPM sends one **1** for L information bits, which would otherwise require an average of L/2 **1**'s for an OOK (no coding) based system. So R_p for L-ary PPM is L/2, which is greater than 1 except when L=2.

If the channel is the AWGN channel, we can calculate the energy required to send the uncoded sequence ($E_{a,u}$) and the coded sequence ($E_{a,c}$) by

$$E_{a,u} = K_a\cdot[W_{a,u}\cdot E_{a,u,1} + (1-W_{a,u})\cdot E_{a,u,0}] \tag{27}$$

and $$E_{c,u} = N_a \cdot [W_{a,c} \cdot E_{a,c,1} + (1-W_{a,c}) \cdot E_{a,c,0}] \tag{28}$$

where K_a is the number of information bits per word, N_a is the number of channel bits per word, $W_{a,u}$ is the fractional weight of the uncoded information sequence, $W_{a,c}$ is the fractional weight of the coded channel sequence, $E_{a,u,0}$ is the energy required to send an uncoded **0** over the AWGN channel, $E_{a,u,1}$ is the energy required to send an uncoded **1** over the AWGN channel, $E_{a,c,0}$ is the energy required to send a coded channel **0** over the AWGN channel, and $E_{a,c,1}$ is the energy required to send a coded channel **1** over the AWGN channel. With OOK signalling, we do not use energy for the signal **0**, so $E_{a,u,0} = E_{a,c,0} = 0$. Then we have

$$E_{a,u}/E_{a,c} = (K_a\ W_u\ E_{a,u,1})/(N_a\ W_c\ E_{a,c,1}) = R_p\ (E_{a,u,1}/E_{a,c,1}) \tag{29}$$

Combining (29) with (1) and the definition of CBVR, we have

$$E_{a,u}/E_{a,c} = R_p\ (S_{a,u}/S_{a,c})^2 = R_p\ (\mathrm{CBVR}_a)^2 \tag{30}$$

where CBVR_a is the CBVR for the AWGN channel.

If instead we consider the DDO channel, we have a similar result,

$$E_{d,u}/E_{d,c} = R_p \cdot \mathrm{CBVR}_d \tag{31}$$

where CBVR_d is the CBVR for the DDO channel. Notice that for R_p, we do not distinguish between the two channels, because if we use the same code, R_p is identical for them. Also notice that if the DDO channel has F'=0, then for the same code, CBVR_d will be the same as CBVR_a.

Now we can compare the CMG of the AWGN and DDO channel used with the same code. First we compare the AWGN channel with the DDO channel when F'=0. Then we compare the AWGN channel with the DDO channel with F'>0. From (30) and (31), we can write the CMG of the channels as

$$\begin{aligned} \mathrm{CMG}_a &= 10\ \log_{10} R_p (\mathrm{CBVR}_a)^2 \\ &= 10\ \log_{10} R_p + 20\ \log_{10} \mathrm{CBVR}_a \\ &= G_r + 2G_s \end{aligned} \tag{32}$$

and $$\begin{aligned} \mathrm{CMG}_d &= 10\ \log_{10} R_p (\mathrm{CBVR}_d) \\ &= 10\ \log_{10} R_p + 10\ \log_{10} \mathrm{CBVR}_d \\ &= G_r + G_s' \end{aligned} \tag{33}$$

Case 1. **F'=0**. When F'=0 for the DDO channel, we have $G_s'=G_s$, because $\mathrm{CBVR}_a=\mathrm{CBVR}_d$. Then we have

$$\mathrm{CMG}_d = \mathrm{CMG}_a - G_s = 0.5 \cdot (\mathrm{CMG}_a + G_r) \tag{34}$$

As we showed in the last section, for conventional codes, CBVR > 1 for small p_b, so in turn, $G_s>0$ for small p_b. This indicates that when $G_s > 0$, the CMG of a DDO channel is always less than the CMG of an AWGN channel if they use the same code and/or modulation. On the other hand, if we want to have $\mathrm{CMG}_d > \mathrm{CMG}_a$, we require $G_s < 0$. If we also want $\mathrm{CMG}_d > 0$ (which is usually the purpose), than we have to have $G_r > -G_s > 0$, or $R_p > 1$. So we have proved the following.

Lemma 1. For an F'=0 DDO channel and an AWGN channel with the same code, the R_p of the code must be greater than 1 if CMG_d is positive and greater than CMG_a.

Case2. **F' >0**. First we compare a DDO channel with F'=0 to another DDO channel with F'>0. When invariant codes are used, the fractional weight of the code sequences generated is 0.5, assuming the fractional weight of the information sequences is 0.5. Then the average channel noise is the average of the noises generated when **0** and **1** are sent. Since we have shown that the CBVR>1 (for interesting error rates) for invariant codes, we use less signal strength to send a channel **1** than we would without a code. When F'>0, the noise variance when a channel **1** is sent decreases as the signal strength decreases, yielding a lower average channel noise level. As a result, for the same error rate, the required signal strength can be even smaller then it would be if the noise was not signal dependent. The effect is more dramatic as F' increases since the constant of proportionality between the noise variance and the signal strength is even larger (see(11)). In other words, the gain due to the coding/modulation increases as F' increases. So we have proved the following.

Lemma 2. If an invariant code is used on a DDO channel, the CMG of a channel with F'>0 is larger than that of a channel with F'=0.

For codes that are not invariant, the situation is more complicated. However, in the case of PPM, we can at least show numerically that Lemma 2 still holds. Some examples will be presented shortly.

Combining Lemmas 1 and 2, we have the following theorem.

Theorem 3. If a DDO channel has a larger CMG than that of an AWGN channel using the same code, the R_p of the code must be greater than 1.

The converse of the theorem is certainly true, i.e., if $R_p<1$, a DDO channel always has a smaller CMG then that of an AWGN channel with the same code (see (34)).

From Theorem 3, we see that the conventional codes (to be more exact, all the invariant codes) would not be very effective if applied to a DDO channel. If there are two different codes which have almost the same CMG when used on an AWGN channel, but have different values of R_p, the one with the higher R_p will out perform the other when applied to a DDO channel. So the theorem also advises us on how to choose good codes and/or modulation schemes for DDO channels.

The CMG's for some sample codes on the AWGN and DDO channels are shown in Figures 3 through 8. In all cases, the channel parameters are normalized to have a mean signal level of 0 and a noise variance $\sigma_0{}^2=1$ when a **0** is sent, and a mean level of S when a channel **1** is sent. In the DDO channel, the noise variance when a channel 1 is sent is $\sigma_1{}^2=1+F'\cdot S$. In each case, the CMG for the AWGN channel and the DDO channels (for various values of F') are plotted against the target bit error rate, p_b. The curves are calculated by first picking some value, S_c, of the signal strength per coded channel **1**. With this signal strength, the bit error probability after coding, p_b, is calculated from the symbol error probability (for PPM) or the codeword error probability (in the case of block or convolutional codes). Then a value of the uncoded signal strength per bit, S_u, is picked, ε and δ are calculated using (2), and $p_b=(\varepsilon+\delta)/2$ since we assume 0's and 1's are sent with equal probability. (The value of T that minimizes p_b is the abscissa corresponding to the intersection point of $p(y|1)$ and $p(y|0)$.) This process is repeated until the p_b so obtained matches that of the coded values. The corresponding values of S_u and S_c are used in (32) and (33) to calculate the CMG for the specified codes.

In Figure 3, the CMG for 2-ary (4 slots) PPM is plotted as a function of the desired bit error rate for the AWGN channel, and the corresponding DDO channels for F'=0 and F'=1.5. As can be seen from the figure, the CMG for F'=1.5 is significantly higher than the values for F'=0 and for the AWGN channel. In Figure 4, the CMG is plotted for 6-ary PPM. Hard decision, "errors only" decoding was used on all the channels. It is important to note that for each PPM example, the AWGN channel showed the lowest gain, and the larger F' DDO channels offered the highest CMG's. In Figure 5, a (7,4) Hamming code is used to illustrate the fact that not all block codes perform well on the optical channel. The Hamming code performs much better on the AWGN channel, and shows its poorest performance on the F'=0 DDO channel. Even with Hamming codes, however, the more non-additive (larger F') optical channels fare better than the more symmetric (lower F') ones. Figure 6 alerts us to the interesting fact that the CMG depends on many factors, one of them being the desired bit error rate of the channel. Although the general trend (AWGN being the best, F'=0 being the worst) seems to hold true for the lower values of p_b, we see that above $p_b=0.06$ (very noisy channels), the (31,21) BCH code performs better on the F'=0 DDO channel than on the AWGN channel! We point out the CMG's for the (7,4) Hamming code and the (31,21) BCH code are quite small (<1 dB for p_b from 0.0001 to 0.1) compared to the gains for 6-ary PPM on the same channels. This seems quite reasonable since the PPM schemes are buying performance with bandwidth. A (2,1,2) convolutional code using hard decision Viterbi decoding is applied to the different channels in Figure 7. The CMG's obtained using this code are similar to those obtained with the (31,21) BCH code in Figure 6. The (31,21) BCH code showed better performance than the (7,4) Hamming code, mainly from its increased block size. Although the (2,1,2) code is not as large as the (31,21) code, its lower rate allows it to compete favorably. As we have seen in all of the examples, the larger F' channels offer higher CMG's than the optical channels with lower values of F', and for conventional codes ($R_p<1$), the CMG on the AWGN channel generally exceeds that on the optical channel. Only when PPM ($R_p>1$) is used, does the CMG on the optical channel exceed that on the AWGN channel. It is also important to note that it is difficult to obtain positive CMG on the optical channel when small block or convolutional codes are used.

V. MULTI-PULSE PPM

PPM has a low code rate, $R=L2^{-L}$, but has a high code-pulse rate, $R_p=L/2$, especially for large L. Actually, as PPM only sends one channel 1 in a word, it maximizes the R_p, so it maximizes G_r. From (34), we see that if a code with $G_r > 0$) is applied to both the AWGN channel and to the DDO channel, the CMG on the optical channel will be higher. This clearly explain why PPM is used on DDO channels. However, because of its low rate, the cost paid for the higher performance is a large bandwidth. The bandwidth expansion factor, γ is the inverse of the code rate, i.e. $\gamma=2^L/L$. For PPM, as G_s' is usually less than 0, $CMG_d < G_r$ from (33). The only way to obtain a large CMG is to have a large L. The consequence is that we have to use a large bandwidth. For example, if we want the CMG_d to be close to 6 dB, then R_p needs to be at least 4, that is L has to be at least 8, and that will require a bandwidth of at least 256/8=32 times larger than that used for the uncoded system.

We strive to obtain a high R_p system which uses as little bandwidth as possible. We cannot obtain a rate as high as PPM's, but we can come close. If we lower the code-pulse rate by sending two pulses for each codeword (instead of one), we decrease G_r, but we also decrease the bandwidth requirement substantially. We call this coding/modulation strategy 2 Pulse PPM, or 2P-PPM. Since two pulses in x time slots can produce $x(x-1)/2$ distinct codewords, we require $x(x-1)/2 \geq 2^L$ to encode blocks of L information bits.

The rate of 2P-PPM is $R=L/x$, so the bandwidth expansion factor is $\gamma=x/L$. For each L information bits, we send two channel 1's (pulses), so the code-pulse rate is $R_p=L/4$. Compared with PPM, we see that G_r is 3 dB less because R_p is 2 time smaller. This will result in about 3 dB less CMG. The advantage is that 2P-PPM uses bandwidth much more efficiently than PPM. The bandwidth expansion factor, γ, for different values of L for 2P-PPM is shown in Table 2, along with those for PPM. We see clearly that when L is large, 2P-PPM consumes much less bandwidth than PPM.

It is of interest to compare the CMG of 2P-PPM with that of PPM. Shown in Figure 8 is the CMG for a 2P-PPM system with L=6. The decoding rule of the 2P-PPM used here is again "errors only", based on hard decision. We can obtain a word error formula similar to that derived for PPM, and also use the same numerical method to search for the threshold that minimizes the error rate. Comparing the curves in Figure 8 with the curves in Figure 4, we see that the CMG is about 3 dB smaller for 2P-PPM, but we only require a bandwidth one fifth as large. For larger L, the decrease in bandwidth is even larger. In general, if two PPM type codes have the same R_p, their CMG will be similar, since the G_r term in (33) dominates G_s'. While bandwidth is indeed an important parameter, there are applications in which an energy or power constraint exists. How does 2P-PPM perform under these constraints? If we design the 2P-PPM system to have the same CMG as PPM (same R_p), we would use 2L-ary 2P-PPM since for 2L-ary 2P-PPM, $R_p = 2L/2/2 = L/2$, which is equal to R_p for L-ary PPM. For large L, this requires about $x \approx (2^{2L+1})^{1/2}$ time slots, and yields a bandwidth expansion factor, $\gamma \approx 0.7(2^L/L)$. So the same CMG can be obtained with 2P-PPM as with PPM with a relative bandwidth saving of 70%. The price is a slight increase in complexity.

When the bandwidth constraint is even more serious, we can use more than two pulses to encode an L bit information word. Using more than two pulses per word further reduces the bandwidth requirement, but of course involves a larger power penalty. Given the system requirements, a tradeoff must be performed to find an acceptable balance between the bandwidth (2P-PPM, 3P-PPM, etc.) and the power required to obtain the desired bit error rate.

REFERENCES

[1] D. L. Snyder and I. B. Rhodes, "Some Implications of the Cutoff Rate Criterion for Coded Direct-Detection Optical Communication Systems," *IEEE Trans. on Information Theory*, Vol. IT-26, pp. 327-338, May 1980.

[2] R. J. McEliece, "Practical Codes for Photon Communication," *IEEE Trans. on Information Theory*, Vol. IT-27, No.4, pp. 393-397, July 1981.

[3] J. L. Massey, "The How and Why of Channel Coding," Zurich Seminar on Digital Communications, 1983.

[4] M. A. Herro, D. J. Costello, Jr. and L. Hu, "Cutoff Rate Calculations for the Outer Channel in a Concatenated Coding System," *Proceeding of the 1984 Conference on Information Sciences and Systems*, Princeton, New Jersey, March 1984.

[5] S. D. Personic, *Optical Fiber Transmission Systems.* Plenum Press, New York, 1981.

[6] S. D. Personic, P. Balaban, J. H. Bosin and P. R. Kumar, "A Detailed Comparison of Four Approaches to the Calculation of the Sensitivity of Optical Fiber System Receivers," *IEEE Trans. on Communications*, Vol. COM-25, pp. 541-548, May 1977.

[7] N. Sorensen, R. Gagliardi, "Performance of Optical Receivers with Avalanche Photo Detection," *IEEE Trans. on Communications*, Vol. COM-27, pp. 1315-1321, September 1978.

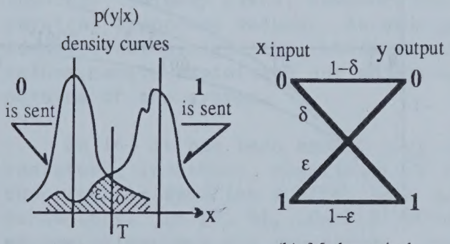
[8] R. M. Gagliardi, G. Prati, "On Gaussian Error Probabilities in Optical Receivers," *IEEE Trans. on Communications*, Vol. COM-28, pp. 1742-1747, September 1980.

[9] J. B. Abshire, "Performance of OOK and Low-Order PPM Modulations in Optical Communications When Using APD-Based Receivers," *IEEE Trans. on Communications*, Vol. COM-32, pp. 1140-1143, October 1984.

[10] S. Lin, D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, New Jersy, 1983.

[11] A. J. Viterbi, "Convolutional Codes and Their Performance in Communication Systems," *IEEE Trans. on Communications*, Vol. COM-19, pp. 751-772, October 1971.

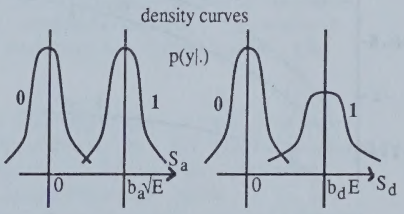
[12] M. A. Herro and L. Hu, "Code Performance for a Direct-Detection Optical Fiber Channel," *Proceeding of 1984 Allerton Conference*, pp. 743-744, Monticello, Illinois, October 1984.



(a) Physical channel model

(b) Mathematical channel model

Figure 1 General channel model



(a) AWGN channel

(b) DDO channel

Figure 2 AWGN and DDO channel model

P_b	0.1	0.01	0.001	0.0001
S_u	2.562	4.653	6.183	7.442
S_c	3.306	5.187	6.616	7.817
CBVR	0.775	0.897	0.935	0.952

Table 1 Comparison cf signal strength needed for a given error rate without and with 2-ary PPM

L	PPM	2P-PPM
4	4	1.75
5	6.4	1.8
6	10.7	2
7	18.3	2.43
8	32	3

Table 2 Comparison of bandwidth expansion factor for PPM and 2P-PPM

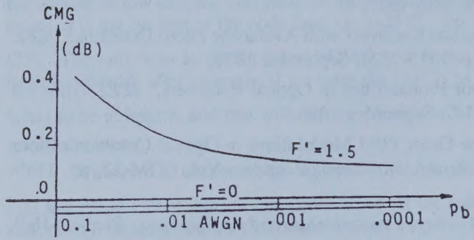


Figure 3 CMG for 2-ary PPM

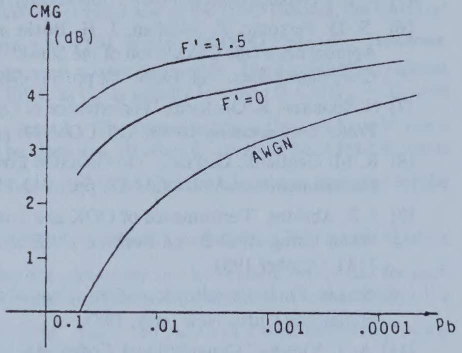


Figure 4 CMG for 6-ary PPM

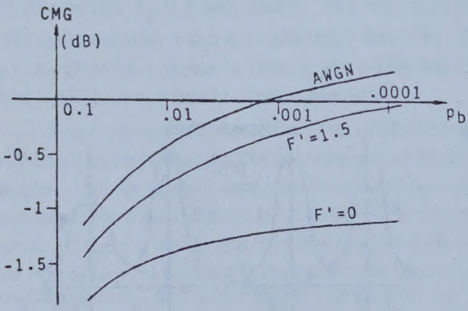


Figure 5 CMG for the (7,4) Hamming code

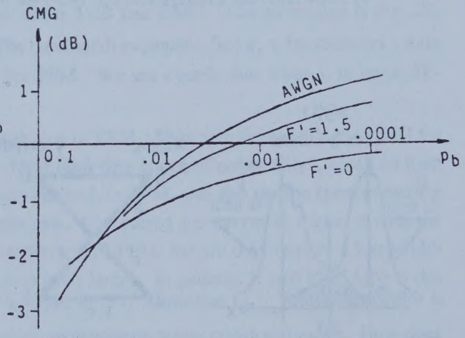


Figure 6 CMG for the (31,21) BCH code

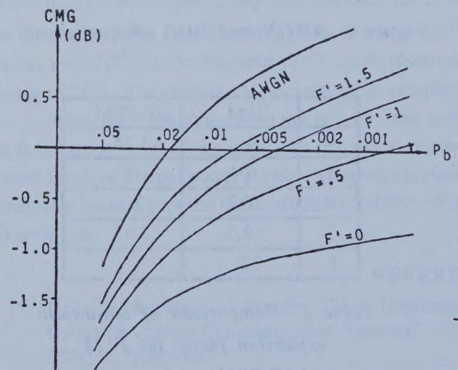


Figure 7 CMG for a (2,1,2) convolutional code

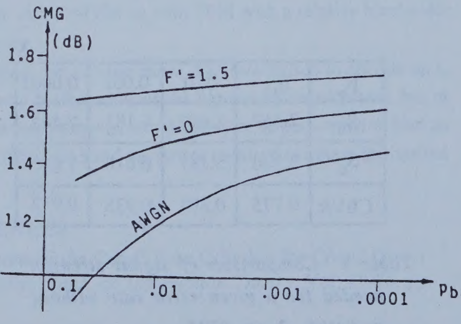


Figure 8 CMG for 6-ary 2P-PPM

ADAPTIVE IDENTIFICATION OF SYSTEMS WITH POLYNOMIAL DEPENDENCE ON UNKNOWN PARAMETERS

SOURA DASGUPTA
Department of Electrical & Computer Engineering, The University of Iowa, Iowa City, Iowa 52242

ABSTRACT

In many physical systems the only unknowns are the physical element values. These, usually appear in the numerator and denominator coefficients of the system transfer function in a polynomial fashion. This paper proposes two algorithms for identifying such system parameters directly. Both algorithms are shown to be uniformly asymptotically convergent on the satisfaction of certain persistence of excitation condition on the input; and in case of the second algorithm when magnitude bounds on the unknown parameters are available. The latter information is readily obtained given the physical significance of these parameters.

1. INTRODUCTION

A number of algorithms for identifying linear time invariant systems exist in the literature [1-5]. In all of these, the only knowledge exploited are the order and relative order of the unknown system. Additional structural knowledge, usually available for physical systems is ignored. In many cases, however, the only unknowns in a system are some physical component values. Accordingly, in this paper, we formulate identification algorithms which directly estimate physical component values, while exploiting any additional information provided by the physics of the system.

In [6] it has been shown, that when the physical parameters are resistors, inductors, capacitors or their mechanical and chemical analogs, the transfer function coefficients are multilinear in the unknown parameters. In [7, 8], identification algorithms for such parameterizations are presented. However, when the unknown parameters include elements like mutual inductors, gear ratios or gyrators, the dependence is in general polynomial.

In this paper we present two equation error algorithms - referred to as two step and single step algorithms - for identifying systems whose transfer function coefficients are polynomial in the unknown parameters. The algorithms parallel in philosophy their counterparts in [7, 8]. Thus, in the two step algorithm, the first step generates an unconstrained estimate of the unknown parameters, which ignores the underlying polynomial dependence. The second step generates an estimate, which takes the nonlinearities into account and which is in some sense the "nearest" to the unconstrained estimate. For the multilinear case [9] simulations show that the second step improves upon the estimate obtained in the first step, a conclusion expected to hold for the general polynomial case as well.

The single step algorithm combines these steps into one, through the use of penalty function ideas. But, unlike the first algorithm, it requires the knowledge of parameter magnitude bounds. Given the physical significance of these parameters, such a knowledge is provided by the laws of physics. Both the algorithms turn out to be uniformly asymptotically stable in the large under persistence of excitation conditions on the inputs.

2. THE ALGORITHMS

Consider the nth order, asymptotically stable linear time-invariant, proper system, with transfer function

$$T(s) = \frac{N(s)}{D(s)} = \frac{p_o(s) + \underset{\sim}{K}^T \underset{\sim}{P}(s)}{q_o(s) + \underset{\sim}{K}^T \underset{\sim}{Q}(s)} \tag{1}$$

input u(t) and output y(t). Here $\underset{\sim}{K}$ is a vector whose first N elements are the unknown parameters k_1,--k_N. The other elements of $\underset{\sim}{K}$ are those polynomial combinations of the form

$$\prod_{i=1}^{N} k_i^{n_i} ,$$

which are present in either N(s) or D(s). $\underset{\sim}{P}(s)$ and $\underset{\sim}{Q}(s)$ are known polynomial vectors with the same dimension as $\underset{\sim}{K}$ and $p_o(s)$ and $q_o(s)$ are known scalar polynomials. Thus if

$$T(s) = \frac{p_o(s) + k_1 p_1(s) + k_2 p_2(s) + k_1 k_2^2 p_5(s) + k_1^3 p_6(s)}{q_o(s) + k_1 q_1(s) + k_2 q_2(s) + k_1^2 q_3(s) + k_1 k_2 q_4(s)} \tag{2}$$

then

$$\begin{aligned} &\underset{\sim}{K} \triangleq \left(k_1,\ k_2,\ k_1^2,\ k_1 k_2\ ,\ k_1 k_2^2 , k_1^3\right)^T \\ &\underset{\sim}{Q}(s) \triangleq \left(q_1(s),\ q_2(s),\ q_3(s),\ q_4(s), 0, 0\right)^T \\ &\underset{\sim}{P} \triangleq \left(p_1(s),\ p_2(s),\ 0, 0,\ p_5(s), p_6(s)\right)^T \end{aligned} \tag{3}$$

Note that we have assumed the presence of the N linear terms $k_1,\ldots k_N$ in either N(s) or D(s). Then (1) can be equivalently written as

$$h_o(t) + \underset{\sim}{K}^T \underset{\sim}{H}(t) = 0 \tag{4}$$

where with $d \triangleq d/dt$ and α any positive number,

$$h_i(t) \triangleq \frac{1}{(d+\alpha)^n} \left(p_i(d)u(t) - q_i(d)y(t)\right) \tag{5}$$

The introduction of $1/(d+\alpha)^n$ is necessitated by the need to avoid explicit differentiation of y(t) and u(t). Our task is essentially one of estimating $\underset{\sim}{K}$ given $h_o(t)$ and $\underset{\sim}{H}(t)$.

Algorithm 1.

This is a two step algorithm. The first step generates an unconstrained estimate $\underset{\sim}{K}u$ of $\underset{\sim}{K}$, where the elements of $\underset{\sim}{K}_u$ may be independent of each other. The second step generates an estimate $\hat{\underset{\sim}{K}}$ "nearest" to $\underset{\sim}{K}u$ with elements obeying the same polynomial interdependence as the elements of $\underset{\sim}{K}$. Thus for (2)

$$\hat{\underset{\sim}{K}} = \left(\hat{k}_1, \hat{k}_2, \hat{k}_1^{\,2}, \hat{k}_1\hat{k}_2, \hat{k}_1\hat{k}_2^{\,2}, \hat{k}_1^{\,3}\right)^T$$

While $\underset{\sim}{K}_u$ is generated by any of the algorithms in [1-5], $\hat{\underset{\sim}{K}}$ is generated by the steepest descent minimization of

$$L_u = (\hat{\underset{\sim}{K}}-\underset{\sim}{K}u)^T \underset{\sim}{\Lambda} (\hat{\underset{\sim}{K}}-\underset{\sim}{K}u) \tag{6}$$

$$\text{with } \underset{\sim}{\Lambda} = \text{diag } \{\lambda_i\}_{i=1}^{\dim(\underset{\sim}{K})}, \ \lambda_i > 0. \tag{7}$$

The overall algorithm could then be

$$\dot{\underset{\sim}{K}}u(t) = -\underset{\sim}{H}(t)\left(\underset{\sim}{H}^T(t)\underset{\sim}{K}_u(t) + h_o(t)\right) \tag{8}$$

$$\dot{\hat{k}}(t) = -\frac{-\partial L_u}{\partial \hat{\underset{\sim}{k}}} = \frac{-\partial \hat{\underset{\sim}{K}}^T}{\partial \hat{\underset{\sim}{k}}} \Lambda\, (\hat{\underset{\sim}{K}}-\underset{\sim}{K}u), \ \hat{\underset{\sim}{k}}(0) = 0 \tag{9}$$

with $\hat{\underset{\sim}{k}} \overset{\Delta}{=} \left(\hat{k}_1, \ldots \hat{k}_N\right)^T$. For assured convergence the following additional constraint is required: if at any t_1 an element of $\underset{\sim}{K}u$ changes sign, $\underset{\sim}{k}(t_1)$ is re-initialized to zero.

Algorithm 2.

Here a single unconstrained estimate $\underset{\sim}{K}_S$ is generated, which through the use of a penalty function is asymptotically constrained to obey the nonlinear constraints. Define

$$L_S = -\frac{\lambda}{2} (\underset{\sim}{K}_S-\underset{\sim}{K})^T(\underset{\sim}{K}_S-\underset{\sim}{K}) + \tfrac{1}{2} f^2(\underset{\sim}{K}_S) \tag{10}$$

with $\lambda > 0$ and

$$f^2(\underset{\sim}{K}_S) = \sum_{j=N+1}^{\dim(K)} \tilde{\lambda}_j f_j^{\,2}(\underset{\sim}{K}_S) \qquad , \ \lambda_j > 0 \tag{11}$$

such that if the jth element of $\underset{\sim}{K}$ is $\prod_{i=1}^{N} k_i^{m_i}$ then

$$f_j(\underset{\sim}{K}_S) = K_{Sj} - \prod_{i=1}^{N} K_{Si}^{\ m_i} \tag{12}$$

Also define

$$\underset{\sim}{R}(t) = \int_0^t e^{-\beta(t-\tau)} \underset{\sim}{H}(\tau)\underset{\sim}{H}^T(\tau)d\tau \tag{13}$$

$$\underset{\sim}{r}_o(t) = \int_o^t e^{-\beta(t-\tau)} ho(\tau)\underset{\sim}{H}(\tau)d\tau \tag{14}$$

Note that both $\underset{\sim}{R}(t)$ and $\underset{\sim}{r}_o(t)$ can be generated on line, [5]. Then the second algorithm becomes

$$\dot{\underset{\sim}{K}}_S = -\lambda\left(\underset{\sim}{R}(t)\underset{\sim}{K}_S + \underset{\sim}{r}(t)\right) - \tfrac{1}{2}\,\underset{\sim}{R}(t)\frac{\partial f^2}{\partial \underset{\sim}{K}_S}\,(\hat{\underset{\sim}{K}}_S) \tag{15}$$

3. CONVERGENCE CONSIDERATIONS

The following theorems summarize the convergence properties.

Theorem 1.

Consider the adaptive law (8 - 9) with the reinitializing condition in place. Then $\left(\underset{\sim}{K}_u^T, \hat{\underset{\sim}{k}}^T\right)^T$ converges uniformly asymptotically to $\left(\underset{\sim}{K}^T, \underset{\sim}{k}^T\right)^T$ with $\underset{\sim}{k} \triangleq \left(k_1, \ldots k_N\right)^T$, provided there exist $\alpha_1, \alpha_2, \delta > 0$ such that $\forall\ t\ \varepsilon\ R_+$

$$\alpha_1\underset{\sim}{I} \leqslant \int_t^{t+\delta} \underset{\sim}{H}(\tau)\underset{\sim}{H}^T(\tau)d\tau \leqslant \alpha_2\underset{\sim}{I} \tag{16}$$

Theorem 2

Consider the adaptive law (11 - 15) with (16) holding. Then $\underset{\sim}{K}_S$ converges uniformly asymptotically to $\underset{\sim}{K}$ if $\forall\ i\varepsilon\{1, \ldots N\}$

$$K_{Si}(t)\ k_i \geqslant 0 \tag{17}$$

Remarks

(1) Equation (17) can be satisfied if magnitude bounds on the k_i are known. This is so as for (15), $L_S(t) \leqslant L_S(0)$. Thus a linear translation in the parameters in the same vein as for [7], can ensure that with appropriately selected $K_{Si}(0)$, $L_S(t) > L_S(0)$ if at any t (17) is violated. The details are omitted.

(2) Equation (16) is a persistence of excitation condition. Under certain conditions it will never hold, no matter what the input. If the following assumptions are satisfied, the u(t) for which (16) holds, can be found:

Assumption 1.

There exists no nonzero vector $\underset{\sim}{\theta}$ such that

$$T(s) = \frac{\underset{\sim}{\theta}^T\underset{\sim}{P}(s)}{\underset{\sim}{\theta}^T\underset{\sim}{Q}(s)} \qquad \forall\ s\ \varepsilon\ C^1$$

Assumption 2

There exists no nonzero vector $\underset{\sim}{\theta}$ such that

$$\underset{\sim}{\theta}^T \underset{\sim}{P}(s) \equiv \underset{\sim}{\theta}^T \underset{\sim}{Q}(s) \equiv 0 \qquad \forall\, s \in C^1$$

Once these are satisfied then the following condition is sufficient to ensure (16): With

$$\underset{\sim}{W}(t) = \left(u(t), \frac{1}{d+\gamma}\, u(t)\ .\ .\ .,\ \frac{1}{(d+\gamma)^m}\, u(t)\right)^T, \ \gamma > 0$$

and $m \triangleq$ highest degree among the polynomials

$p_o(s)q_i(s) - q_o(s)p_i(s)$, $\forall i \varepsilon \{1, .\ .\ N\}$ and $p_i(s)q_j(s) - q_i(s)\ p_j(s)$

$\forall i,j \varepsilon \{1, .\ .\ N\}$, there exist γ_1 , γ_2, $\delta' > 0$

$$\gamma_1 \underset{\sim}{I} \leqslant \int_t^{t+\delta'} \underset{\sim}{W}(t)\underset{\sim}{W}^T(t)dt \leqslant \gamma_2 \underset{\sim}{I}$$

Remark: The condition essentially requires that if u(t) is a linear combination of sinusoids then it must contain at least m distinct frequency components.

4. CONCLUSION

Two adaptive identification algorithms are proposed for systems whose numerator and denominator transfer function coefficients have a polynomial dependence on the unknown parameters. The second algorithm requires information regarding the magnitude bounds of the unknown parameters for uniform asymptotic convergence. In addition both require that the input be persistently exciting in a sense described in this paper.

REFERENCES

[1] P.M. Lion, "Rapid Identification of Linear and Nonlinear Systems," AIAA Journal, Vol. 5, Oct. 1967, pp. 1835 - 1842.

[2] K.S. Narendra and P. Kudva, "Stable Adaptive Schemes for System Identification and Control - Parts I and II," IEEE Trans. on Syst. Man. Cyber, Vol. SMC-4, pp. 542-560, Nov. 1974.

[3] R.L. Carroll and D.P. Lindorff, "An Adaptive Observer for Single Input-Single Output Linear Systems," IEEE Trans. on Auto Contro., AC-8, No. 5, pp. 428 - 435, 1973.

[4] B.D.O. Anderson, "An Approach to Multivariable System Identification," Automatica, Vol. 13, pp. 401 - 408,1977.

[5] G. Kreisselmeier, "Adaptive Observers With Exponential Rates of Convergence," IEEE Trans. on Auto Contr, AC-22, Feb. 1977, pp. 2-8.

[6] B.D.O. Anderson and S. Dasgupta, "Physically Based Parameterizations For Designing Adaptive Algorithms," Under preparation.

[7] S. Dasgupta, B.D.O. Anderson and R.J. Kaye, "Identification of Physical Parameters in Structured Systems," Submitted to Automatica.

[8] S. Dasgupta, B.D.O. Anderson and R.J. Kaye, "Output Error Identification Methods for Partially Known Systems," To Be Published in Int. J. of Control.

[9] S. Dasgupta, "Adaptive Identification and Control," Ph.D. Thesis, Autralian National University, Canberra, Australia, 1984.

EFFECTS OF VARIABLE LOADS ON GROSS MOTION OF ADAPTIVELY CONTROLLED MANIPULATOR

SHAHRAM ZAREI* and A. J. KOIVO
Electrical Engineering Dept., Purdue University, W. Lafayette, IN
*Now employed with GM Research Laboratories, Warren, MI

ABSTRACT

Variable loads affect the manipulator motion causing difficulties in the control. Changes in the dynamics and control of the manipulator are examined here due to the weight of an object attached to the gripper of the last link. A simulation study is performed with the mass of the load ranged from zero up to 1/8 of the manipulator mass.

I. PROBLEM STATEMENT

The principal axes of a load with homogeneous mass, while being grasped by the gripper, are aligned with the coordinate system of the last link, Figure 1. M and m are assumed to be the masses of the last link and load, respectively, and X_0, Y_0, Z_0 are the components of the center of the mass of the last link with respect to its coordinate system. The components of the new center of mass for the last link are as follows:

$$Z_1 = (M \cdot Z_0 + m \cdot Z)/M_1;\ X_1 = X_0;\ Y_1 = Y_0$$

where Z is the center of the mass component of the load with respect to the last link coordinates, and M_1 is the total value of mass of the last link including the load mass: $M_1 = M + m$.

Knowing the moment of inertia of the load, one can refer the inertia of load to the last link coordinate frame, according to the parallel axis theorem [1], and add it to the old inertia of the link:

$$I_1 = I_{10} + I + (m \cdot r^2) \cdot I_{den}$$

$$I_1 = [I_{1x}, I_{1y}, I_{1z}];\ I_{1o} = [I_{1ox}, I_{1oy}, I_{1oz}];\ I_{den} = [1.0, 1.0, 1.0]$$

where I_1 is the modified value for the inertia of the last link, vector I_{1o} contains the old value for the inertia of the last link, vector I consists of the moments of inertia of the load with respect to the x,y, and Z axes, r is the distance between the mass center of the load and the last link, and I_{den} is an identity vector. Thus, the above equation is used to make necessary changes in the pseudo-inertia matrix of the last link.

II. SIMULATION RESULTS

An object is assumed to be picked up by the gripper of the Puma manipulator. The task is to move the gripper with the load from point P_1 to point P_2 in 0.8 seconds. The path of the gripper is a straight line in the Cartesian coordinate system. It is desired that the speed of the end effector follows a trapezoid starting at a constant acceleration from the zero initial speed until it reaches a maximum speed of 28.1 in/s in 0.1s, holds this maximum speed for 0.6s, and the decelerates to zero speed at the terminal state P_2. The numerical values chosen for P_1 and P_2 were $P_1 = (10.5, 4, 30)$ and $P_2 = (30, 14, 35)$ expressed in inches.

In this simulation study, the independent joint control is assumed. The model for each selected joint is:

$$V_i(k) = h_i + a_i V_i(k-1) + b_i u_i(k-1) + e_i(k)$$

where $v_i(\ )$ is the angular velocity of joint i, $u_i(\ )$ is the voltage applied to the actuator of joint i, e_i is the modeling error; and the parameters h_i, a_i and b_i are to be estimated. The adaptive self-tuning type controller used for the separate joint control in this simulation study is described in [2].

The simulation results for the manipulator, while carrying the load of the different masses, with the adaptive controller are shown in Figures 2 and 3 for the position of joint 4 to joint 6 in the joint coordinates.

REFERENCES

[1] F. P. Beer and E. R. Johnston, Mechanics for Engineers (Statics), McGraw Hill, 1976 pp. 353-354.

[2] A. J. Koivo and T. H. Guo, "Adaptive Linear Controller for Robotic Manipulators", IEEE Trans. on Automatic Control Vol. AC-28, No. 2, February 1983, pp. 162-171.

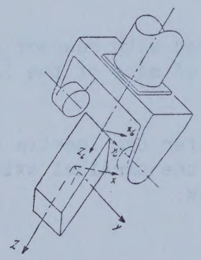


Figure 1. Position of the Load with Respect to Gripper

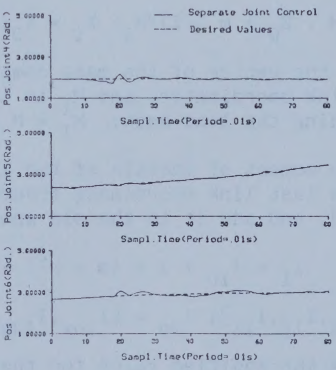


Figure 2. Simulation Results of the Position of Joint 4 to Joint 6 (Load's Mass=5)

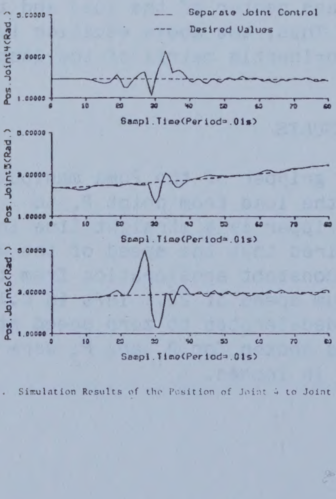


Figure 3. Simulation Results of the Position of Joint 4 to Joint 6 (Load's Mass=20)

INVESTIGATIONS INTO THE ROBUSTNESS OF SELF-TUNING CONTROL*

KETAO LIU, JANOS GERTLER
GEORGE MASON UNIVERSITY
FAIRFAX, VA 22030

Robustness of Self-tuning Controller is discussed. A supervisory scheme to guard closed-loop stability is presented.

Robustness of Self-tuning Controller has been one of the topics of recent control research. While a number of control schemes were proved to work properly in theoretical studies, practical investigations have been shown that they may fail under non-ideal circumstances. The fundamental problem is the loss of long-term stability, attributed mostly to the structural discrepancy between the plant model and the real plant (Kokotovic, 1985), and to the lack of sufficient excitation for the identifier to track the change of plant dynamics in steady state (Anderson , 1985).

Our investigations was concentrated on the effect on stability of the controller design methodology and of identification errors. We found that if the controller is designed to optimize a quadratic performance index, the induction of control terms in the index generally improves stability but does not guarantee it. On the other hand, pole-placement methods keep track of stability easily but are inflexible with respect to identification errors.

While general theoretical guarantees are not available, several authors recommend practical schemes to tackle the problem of robustness. Our approach is an extension to the supervisor proposed by Isermann and Lachmann (1982) and is related to the "expert control" described by Astrom (1985). In addition to the basic control loop (plant, controller and feedback) and the adaptive loop (identifier and tuner), our scheme contains a supervisory level, consisting of an instability indicator, a signal generator, a structure identifier and a detuner (Figure 1.). When instability is detected, the controller parameters are detuned to safer values and the structure and parameters of the model are re-identified; then the controller is returned based on the new model.

Instability detection is obtained in a two-stage on-line analysis of the plant output. When the first stage, watching the short-term average of the squared control error, indicates abnormality, a fast and simple identifier is triggered to distinguish between transients and instability. The controller is detuned by bringing all controller poles and zeros closed to the origin (in the z-domain) and reducing the loop-gain simultaneously; any loss in the closed-loop steady-state gain is compensated for outside the loop. To seek a new plant model identifier with several model structures (different model orders and dead-times in the neighborhood of the most recent model) is run in parallel and the best one is selected on the basis of the performance index.

Simulation experiments show good performance with the instability detector and detuner connected on line. Work to develop a fast structure identifier is under way.

---

* This work was supported by NSF under grant ECS-8400765

References:
Anderson, B.D.O. (1985) Automatica, 21, pp. 247-258
Astrom, K.S. (1980) American Control Conference, pp. 1514-1519
Isermann, R.; Lachmann, K.H. (1982) 3rd IFAC-IFID Symposium on Software for Computer Control, pp. 25-49
Rohrs, C.L. et al. (1981) 20th IEEE Conference on Decision and Control, pp. 1272-1284
Rohrs, C.L. et al. (1982) 21th IEEE Conference on Decision and Control, pp. 3-11
Kokotovic, P. (1985) Automatica, 21, pp.225-236

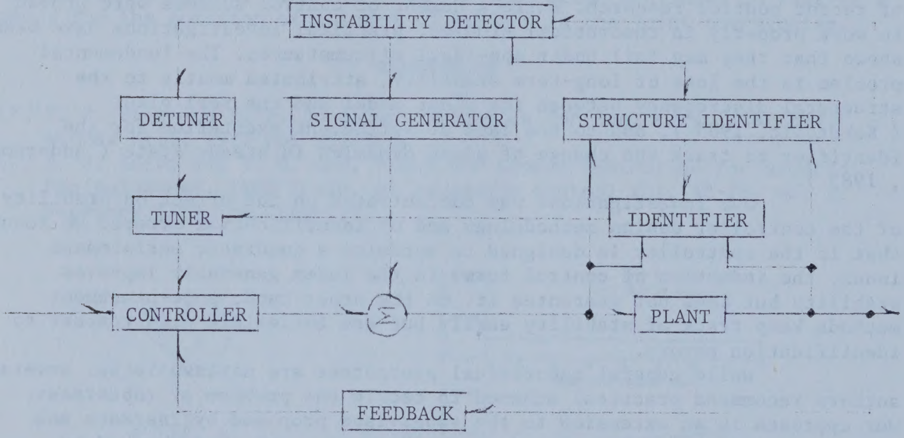

INSTABILITY DETECTOR
DETUNER
SIGNAL GENERATOR
STRUCTURE IDENTIFIER
TUNER
IDENTIFIER
CONTROLLER
Σ
PLANT
FEEDBACK

Figure 1.

ADAPTIVE FORCE-POSITION-VELOCITY CONTROL FOR ROBOTIC MANIPULATORS

A. J. Koivo
School of Electrical Engineering
Purdue University
West Lafayette, IN 467907

Abstract

The force-path control of a robotic manipulator is presented here in the joint and in the Cartesian coordinate systems. An autoregressive model with external excitation (ARX-model) is introduced for designing an adaptive controller with self-tuning. The controller minimizes the conditional expectation of the sum of a quadratic position (velocity) error and a quadratic force error while satisfying the constraint of the ARX-model with the estimates substituted for the unknown parameters. The basic approach is used to obtain an adaptive controller which operates on the variables expressed in the joint coordinates.

I. Introduction

When a robotic manipulator is used to perform complex material handling and assembly tasks, sensors (transducers) can provide useful information about the environment. On the basis of this information, the robotic manipulator can interact with the environment, and adapt to changes and uncertainties encountered during the operation. For example, errors in the measurements and/or design specifications in the parts to be assembled can cause difficulties in the completion of a task if there is no feedback from the environment. This problem may partly be circumvented by using sensory (pressure, force, touch, vision) information in the control scheme. An intelligent robot must then be able to adapt its operation according to the constraints imposed by the environment. It can be accomplished by means of sensory feedback loops. One type of sensory information is available from force sensors. In performing an assembly task, it allows the manipulator to compensate for small errors in the measurements of the pieces, the jigs and grippers which hold them. In such a case, the manipulator should be able to change the mode of operation, for example, from the force servoing to position servoing. When this is possible, the manipulator controller can comply with the external constraints; this mode of action is termed the compliance (force) control. A compliant motion control can be used to improve the performance of a manipulator.

In order to use force/torque control with a feedback, it is necessary to obtain measurements of the generalized forces, (i.e., forces/torques) which are to be con-

---

Supported in part by Engineering Research Center on Intelligent Manufacturing funded by National Science Foundation.

trolled. Probably the simplest form of force feedback is based on binary relay type force information, i.e., on contact. The manipulator can be made to stop the motion in the direction when contact is detected, while the motion may be continued in the other directions. In such a system, the motions must take place at slow speeds. A more attractive alternative is to sense the exerted forces/torques continuously. An approximate measure of the generalized forces can be obtained in electric manipulators by measuring the currents of the driving motors; then, a conversion algorithm can be used to calculate the corresponding forces/torques of interest [3]. The generalized forces acting on the robot joints can directly be measured by means of strain gauges. Experimental work on a single-joint manipulator with a force sensor on the joint has been reported [8], and on a Stanford arm with torque sensors on the first two joints in [10]. An alternative and often preferred approach is to measure forces continuously on the wrist of a manipulator; this has been proposed in [1]. Wrist-mounted force sensors are more accurate, and easy to use [5].

The differences between the actual values and the desired values of the generalized forces may be used to design a controller for force servoing. Such an error vector can be formed in the world coordinates and then converted into joint error by means of a Jacobian matrix; it is used in [1] to form a generalized torque vector for the joints. This control scheme compensates also for the torque caused by gravity. Another control scheme with a wrist sensor providing the force information was developed in [3]. The manipulator joint which is most sensitive to compliant motion is used to servo the force, while the position of the gripper is controlled by the other joint motors. The main drawback of the approach is that the force at the beginning of the motion is quite accurate, but at the end it may be quite far from the desired value. It is limited to tasks in which the relative changes due to force servoing are small.

The application of the resolved motion rate control (RMRC) to fine motion makes use of velocity feedback [4]. This approach, called accommodation, controls a force vector; it is essentially a damping control in which a force feedback matrix embodies different control strategies. The method is flexible in different tasks, as is demonstrated by the experiments [4]. A control scheme called resolved motion force control [8] represents a modification to the RMRC-control; it is designed for position control, and not directly for force control.

An alternative approach [6] to force control is to form a position vector from measured forces using a stiffness matrix, then a positional error vector in the Cartesian coordinates, and express it in the joint coordinate system, using the inverse Jacobian matrix. Thus, a corrective force vector to be applied to the joints [6] is obtained. Another approach [5] is to form an error vector of the Cartesian position variables, and multiply it by a compliance matrix; the resulting force vector is compared with the desired value, and the force error vector is converted to a differential torque. Another control loop forms a position error vector in the Cartesian space; it is then converted to a joint error vector using the inverse Jacobian matrix. The goal in [5] is to drive each actuator according to the sum of contributions from the position and force servoing terms, depending on their relative importance. A compliance selection matrix specifies which joints are under position servoing and which are under force servoing.

The aforementioned approaches have different attractive features, and some disadvantages. One of the main problems is due to the calculation time involved, and the need for empirical adjustment (tuning) of the PID-controller gains and parameters, particularly when the task of the manipulator changes. No optimality criterion is used in the design of the controllers mentioned above. The tuning criterion is usually based on the step response of the system.

II. Design Specifications

For a given task of a manipulator, certain position (velocity) and force constraints can be specified. The desired values of these variables can be determined systematically by means of mechanical and geometric characteristics of the given task configuration. It may be convenient to decompose the constraint into natural and artificial constraints [7] to help in defining the desired generalized forces and positions.

The problem of force-position-velocity control will be studied in the joint coordinate system.

III. Manipulator Control in Joint Coordinate System

When both the forces and angular positions are measured in the joint coordinate system, then using the variables in the joint coordinate system for the force-path control is an attractive choice. The model and the performance criterion for the controller design are expressed in the joint coordinate system. The design of an adaptive controller for given design specifications is first presented.

III.1. Problem Statement in Joint Coordinate System

The desired values of the forces (torques), positions, and velocities are assumed to be defined first in the joint coordinate system; i.e., the discrete values $\{q^d(k)\}_k$, and $\{\tau^d(k)\}_k$ represent the desired values for the angular position, velocities, and the generalized force in the joint coordinates. The dynamics of an m-joint manipulator in the joint coordinates will be assumed to be governed by an autoregressive model with external excitation (ARX):

$$q(k) = h + A(z^{-1})q(k) + B(z^{-1})u(k-d) + e(k) \tag{1}$$

where $u(k) = [u_1(k), \ldots, u_m(k)]^T$ signifies the input vector, $u_i(k)$ is the generalized force produced by the i^{th} actuator at time k, and d is a positive integer; $q(k) = [\Theta_1(k), \ldots, \Theta_m(k); \xi_1(k), \ldots, \xi_m(k)]^T$ is the state vector with $\Theta_i(k)$, and $\xi_i(k)$ representing the angular position and velocity of the i^{th} joint, respectively. The vector $h = [h_1, \ldots, h_m]$ corresponds to gravitational forces; e(k) represents a zero-mean Gaussian white noise process with a finite variance; and z^{-1} is a delay operator: $z^{-1}y(k) = y(k-1)$. The matrix polynomials $A(z^{-1})$ and $B(z^{-1})$ are specified by the following expressions:

$$A(z^{-1}) = A_1 z^{-1} + A_2 z^{-2} + \cdots + A_n z^{-n} \tag{2}$$

$$B(z^{-1}) = B_0 + B_1 z^{-1} + \cdots + B_{n-1} z^{-n+1} \tag{3}$$

where the matrices A_i, and B_i, $i = 1,\ldots,n$ contain unknown parameters, which along with the vector h are to be estimated, and n specifies the order of the ARX-model.

In order to meet the design specifications in the joint coordinate system, an adaptive controller will be designed so as to minimize the conditional expectation $E\{\cdot|\,\psi(k)\}$ of the function given in the following expression:

$$I^k[u] = E\{||q(k+d)-q^d(k+d)||^2_{Q(k+d)} + ||u(k)-\tau^d(k)||^2_{R(k)} / \psi(k)\} \tag{4}$$

while satisfying the model equation (1) when the estimated parameters are substituted for the unknowns. The parameter estimates will be calculated on-line and recursively from the well-known difference equations on the basis of the measurements of $\{q(k)\}_k$, and $\{u(k)\}_k = \{\tau(k)\}_k$ where $u(k) = \tau(k)$ is assumed to be available from strain gauges on the joints. The equations for the parameter estimator and adaptive controller are presented next.

III.2. Adaptive Controller with Self-Tuning for Force-Position-Velocity Control in Joint Coordinate System

a. Parameter Estimator Equations

The controller will be designed on the basis of the ARX-model given in equation (1) with the estimates substituted for the unknown parameter values. The equations for calculating the estimated values are given next. The matrix α and the vector M are defined as follows:

$$\alpha = [A_1, \ldots, A_n; B_0, \ldots, B_{n-1}; h]^T = [\alpha_1, \ldots, \alpha_m] \tag{5}$$

where the superscript denotes the transposition. Then for $i = 1,\ldots,m$,

$$\alpha_i = [a^1_{i1}, \ldots, a^1_{i2m}, a^2_{i1}, \ldots, a^2_{i2m}, \ldots; b^0_{i1}, \ldots, b^0_{im}, \ldots, b^{n-1}_{i1}, \ldots, b^{n-1}_{im}; h_i]^T \tag{6}$$

$$M(k-1) = [q^T(k-1), \ldots, q^T(k-n); u^T(k-d), \ldots, u^T(k-d-n+1); 1]^T. \tag{7}$$

Equation (1) can be rewritten as follows:

$$q(k) = \alpha^T M(k-1) + e(k). \tag{8}$$

The algorithm for the parameter estimation is constructed so that one vector at a time is estimated in α of (5). The error criterion is chosen for each vector α_i of α as follows:

$$E(\alpha_i) = \frac{1}{N+1} \sum_{k=n}^{N+n} e_i^2(k) \tag{9}$$

where $(N+n)$ represents the number of measurements, and $e_i(k)$ is the ith component of the e vector. The problem is to minimize $E(\alpha_i)$ relative to the parameter vector α_i.

The solution to the least-squares problem is furnished by the following recursive equations where the caret refers to estimated values:

$$\hat{\alpha}_i(k) = \hat{\alpha}_i(k-1) + P(k)M(k-1)\cdot[q_i(k) - \hat{\alpha}_i^T(k-1)M(k-1)] \tag{10}$$

$$P(k) = \frac{1}{\mu}\left[P(k-1) - \frac{P(k-1)M(k-1)M^T(k-1)P(k-1)}{\mu + M^T(k-1)P(k-1)M(k-1)}\right] \tag{11}$$

where $0 < \mu \leq 1$ is a forgetting factor, which can be used to account for an exponential decaying of past data in tracking a slow drift in the parameters. $P(\cdot)$ is a symmetric matrix.

The parameter estimates will be calculated on line using equations (10) and (11). The estimated parameters are then used in the model (1) to determine the feedback gains and the control $u(k)$.

b. Adaptive Controller with Self-Tuning

The adaptive controller for the manipulator will be designed by minimizing $I^k[u]$ in equation (4) while satisfying the ARX-model in equation (1) with the estimates substituted for the unknown parameters.

The minimization of expression (4) leads to the adaptive controller with self-tuning. It is specified by the following equation:

$$R(k)\,[u^*(k) - \tau^d(k)] + \hat{B}_0^T\,Q(k+d)\,[q(k+d/k) - q^d(k+d)] = 0 \tag{12}$$

where $q(k+d/k)$ represents the predicted value of the vector $q(\cdot)$ at time $(k+d)T$ based on the information up to time kT (included). Equation (12) can be solved for the controller output $u^*(k)$.

Example

A special case of the general solution may be studied by assuming that $d=1$, $A(z^{-1}) = A_1 z^{-1}$, and $B(z^{-1}) = B_0$. Letting $u(k) = \tau(k)$, the ARX-model and the performance criterion are now given by

$$q(k) = \hat{h} + \hat{A}_1\,q(k-1) + \hat{B}_0\,\tau(k-1) + e(k) \tag{13}$$

$$I^k[u] = E\{\|q(k+1) - q^d(k+1)\|^2_{Q(k+1)} + \|\tau(k) - \tau^d(k)\|^2_{R(k)}/\psi\} \tag{14}$$

The unknown parameters in equation (13) are estimated on-line recursively by equations (10) and (11). The adaptive controller with self-tuning is then given by equation (15) and (16):

$$R(k)\,[\tau^*(k) - \tau^d(k)] + \hat{B}_0^T\,Q(k+1)\,[q(k+1/k) - q^d(k+1)] = 0 \tag{15}$$

$$q(k+1/k) = \hat{h} + \hat{A}_1\,q(k) + \hat{B}_0\,\tau^*(k) \tag{16}$$

Equations (15) and (16) specify the adaptive controller; thus, the optimal input $\tau^*(k)$ can be determined:

$$\tau^*(k) = \tau^d(k) + [R(k) + \hat{B}_0^T\,Q(k+1)\,\hat{B}_0]^{-1}\hat{B}_0^T Q(k+1)\,[q^d(k+1) - \hat{h} - \hat{A}_1\,q(k)] \tag{17}$$

Equation (17) specifies the torque to be applied to the joints by the actuators.

In view of the implementation of the adaptive controller, the designer should observe that $\{\tau^d(k)\}$, and $\{q^d(k)\}$ can be related by the following equation:

$$q^d(k) = \hat{h} + \hat{A}_1 q^d(k-1) + \hat{B}_0 \tau^d(k-1) \tag{18}$$

Equation (18) follows from equation (13) by substituting the desired values for the variables. Alternatively, the desired torque τ^d can be computed by the computed torque method [3]. Subtracting equation (18) from (13) results in the following equation, where $\Delta q(k) = q(k) - q^d(k)$, and $\Delta\tau(k) = \tau(k) - \tau^d(k)$:

$$\Delta q(k) = \hat{A}_1 \Delta q(k-1) + \hat{B}_0 \Delta\tau(k-1) + e(k) \tag{19}$$

The adaptive self-tuning controller given by equations (15) and (16) may be rewritten:

$$R(k) \Delta\tau^*(k) + \hat{B}_0^T Q(k+1) [\hat{A}_1 \Delta q(k) + \hat{B}_0 \Delta\tau^*(k)] = 0$$

or

$$[R(k) + \hat{B}_0^T Q(k+1)\hat{B}_0]\Delta\tau^*(k) + \hat{B}_0^T Q(k+1)\hat{A}_1\Delta q(k) = 0 \tag{20}$$

Equation (20) reveals that by selecting the weighting matrices R(k) and Q(k) the controller can perform position or force servoing, i.e., $\Delta q(k)$ or $\Delta\tau^*(k)$, respectively, can be forced towards zero. The choice of R(k) and Q(k) will also allow the possibility of the combined servoing. Equation (20) shows also that the controller gains are dependent upon the current estimates of the parameters; thus, the controller gains are changed when the estimates vary.

Conclusions

The design of a self-tuning controller for controlling the forces exerted by a manipulator, and the path (gross-motion) is presented. An autoregression model with external inputs (ARX) is used as the basis in the design.

References

[1] J. L. Nevins and D. E. Whitney, "The Force Vector Assembler Concept," Proc. 1st Int. Conf. Robots Manipulator Systems, Udine, Italy, Springer-Verlag, New York, September 1973.

[2] H. Inoue, "Force Feedback in Precise Assembly Tasks," MIT Artificial Intelligence Laboratory Memo 308, August 1974.

[3] R. Paul and B. Shimano, "Compliance and Control," Proc. of 1976 Joint Automatic Control Conference, San Francisco, CA, July 1976, pp. 694-699.

[4] D. E. Whitney, "Force Feedback Control of Manipulator Fine Motions," Journal of Dynamic Systems, Measurement and Control, ASME Trans., pp. 91-97, June 1977.

[5] M. H. Raibert and J. J. Craig, "Hybrid Position/Force Control of Manipulators," Journal of Dynamic Systems, Measurement and Control, Vol. 102, June 1981, pp. 126-133.

[6] J. K. Salisbury, "Active Stiffness Control of a Manipulator in Cartesian Coordinates," Proc. of the 19th IEEE Conference on Decision and Cont., pp. 95-100, Albuquerque, New Mexico, November 1980.

[7] M. T. Mason, "Compliance and Force Control for Computer Controlled Manipulators," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-11, No. 6, June 1981, pp. 418-432.

[8] C.-H. Wu and R. P. Paul, "Resolved Motion Force Control of Robot Manipulator," IEEE Trans. on Systems, Man and Cybernetics, Vol. SMC-12, No. 3, pp. 266-275, May/June 1982.

[9] M. Vulobratovic and D. Stokic, "Dynamic Control of Manipulators via Load-Feedback," Mechanism and Machine Theory, 17(2), pp. 107-118, 1982.

[10] J. Y. S. Luh, D. W. Fisher, and C. P. R. Paul, "Joint Torque Control by a Direct Feedback for Industrial Robots," IEEE Trans. Automatic Control, AC-28, No. 2, pp. 132-141, February 1983.

[11] W. E. Snyder, Industrial Robots: Computer Interfacing and Control, Prentice-Hall, Inc., 1985, pp. 202-205.

MULTIDIMENSIONAL CODED PSK SYSTEMS USING UNIT-MEMORY TRELLIS CODES

ALAIN LAFANECHERE
Dept. of Electrical &
Computer Engineering
Illinois Inst. of Tech.
Chicago, IL 60616

and

DANIEL J. COSTELLO, JR.
Dept. of Electrical
Engineering
University of Notre Dame
South Bend, IN 46556

In this paper, we investigate the use of unit-memory trellis codes along with multi-dimensional phase modulation to obtain coding gains greater than those achievable with coded two-dimensional modulation. 2L-dimensional MPSK signal sets are obtained by combining L two-dimensional MPSK signals into a single signal. The signals are chosen by a mapping from the output of an (Ln_0, Lk_0) unit-memory trellis encoder into 2L-dimensional MPSK signals. Since the codes have unit-memory, the state-complexity of the decoder is limited to 2^{Lk_0}. Coding gains of up to 1.9 dB over those achievable with coded two-dimensional MPSK modulation have been obtained.

Let V be the number of memory bits in the encoder. In the design of a trellis code, we regard an encoder simply as a finite-state machine with a given number of states (2^V) and of state transitions (2^k from each state). During each modulation interval, k bits are encoded and one signal space vector in 2L dimensions is transmitted. After selecting a suitable trellis state-transition diagram, the task consists of assigning channel signals to the transitions in order to achieve maximum d_{free}.

UM/PUM TRELLIS CODES COMBINED WITH L*QPSK MODULATION

In this case, the Hamming Distance (HD) is proportional to the squared Euclidean Distance (ED2), and maximizing the free ED2 is equivalent to maximizing the free HD. Moreover the coding gains are the same in terms of ED2 or HD. In most applications of (n,k) convolutional codes, n and k are considered relatively prime, i.e., GCD(n,k) = 1. However, (n,k) codes where GCD(n,k) = L, L > 1, can sometimes achieve a larger free HD for a given state complexity V due to the increased flexibility available when coding more bits at the same time [1,2]. The following table presents the most interesting coding gains attainable.

Rate	1/2			1/3		1/4					2/3
k/n	2/4	4/8	5/10	2/6	6/18	2/8	3/12	4/16	5/20	6/24	4/6
V =	1	3	5	1	6	1	2	3	4	6	2
UM/PUM	P	P	U	P	U	P	P	P	P	U	P
G(dB)	1.25	1.25	.51	.79	.28	.58	.79	.90	.51	.79	1.25

Note that a 1.25 dB coding gain can be achieved by using an (8,4) PUM code instead of the (2,1) code with the same V. The output is mapped onto four QPSK signals. The encoder is more complex but it is the only added complexity since the 4*QPSK signals can be generated using the same QPSK modulator four times.

UM/PUM TRELLIS CODES COMBINED WITH 2*(8PSK) MODULATION

For any 8PSK signal $\underline{v}$, let $\hat{\underline{v}}$ be the 8PSK signal at maximum distance (ED2)

This work was supported by NASA Grants NAG 5-234 and NAG 5-557.

from v. So {1,2,3,4,$\hat{1}$,$\hat{2}$,$\hat{3}$,$\hat{4}$} is the set of 8PSK signals. Their possible ED2's are: D0 = 2 - $\sqrt{2}$; D1 = 2; D2 = 4. For example, the 2*(8PSK) signal (3,$\hat{2}$) can be generated with one 8PSK generator used first to generate the signal 3 and then to generate the signal 2. Recalling the work of Ungerboeck [3], mapping by set partitioning follows from successive partitioning into subsets with increasing minimum distance. Ultimately, pairs of signals with maximum distance are obtained. This is the starting point of what may be called Reverse Set Partitioning (RSP). First the pairs are constructed. Then by joining two pairs in a symmetrical and optimal way, the 4-tuples are formed. By the same process, the 8-, 16-, and 32-tuples are created from the 64 signals in 2*(8PSK). In designing trellis codes for different V, we found as expected that no "asymmetrical" partitions are good.

As an example, consider a rate 4/6 code with 2*(8PSK), partitioned by RSP. The reference system is a rate 2/3 code with the same V, and with 8PSK. The best d_{free} values attainable with this reference system are shown in the table below. In the case V = 4, the gain is 1.895 dB, but the code is catastrophic.

V	8PSK System	2*(8PSK) System	Gain in dB
1	D1 + D0	D1 + 2*D0	0.887
2	D2	D2	0
3	2*D1 + D0	2*D1 + D0	0
4	2*D1 + 2*D0	2*D2	1.895

UM/PUM TRELLIS CODES COMBINED WITH L*(8PSK) MODULATION, L > 2

We also consider rates 2L/3L for L = 3 and 4 and restrict ourselves to a two-state encoder (V=1). The reference system stays the same. For L = 3, the best code has $(d_{free})^2$ = Min{D2, D1 + 3*D0} = D1 + 3*D0, and the coding gain is 1.62 dB over the reference system. For L = 4, the best code has $(d_{free})^2$= = Min{D2, D1 + 4*D0} = D2. The coding gain is 1.895 dB over the reference system. We find that the parallel transitions limit us to this value for L > 4 and that 1.895 dB is the best coding gain attainable in this case.

CONCLUSION

UM/PUM trellis codes have low state-complexity. The Viterbi algorithm can be used for soft decision ML decoding. No extra memory is required at the receiver. Moreover, they present good coding gains at a fixed SNR over the reference codes, with the same V but with GCD(n,k) = L, L > 1. This requires no increase in the signal-generation hardware, which makes these codes even more interesting.

REFERENCES

[1] L.N. Lee, "Concatenated Coding Systems Employing a Unit-Memory Convolutional Code and a Byte-Oriented Decoding Algorithm", IEEE Trans. Commun., COM-25, pp. 1064-1074, October 1977.

[2] G.S. Lauer, "Some Optimal Partial-Unit-Memory Codes", IEEE Trans. Inform. Th., IT-25, pp. 240-243, March 1979.

[3] G. Ungerboeck, "Channel Coding with Multilevel/Phase Signals", IEEE Trans. Inform. Th., Vol. IT-28, pp. 55-67, January 1982.

VITERBI DECODING BY SYSTOLIC ARRAY

C. Y. CHANG
Electrical Engineering Department
University of California
Los Angeles, CA 90024

K. YAO
Electrical Engineering Department
University of California
Los Angeles, CA 90024

Abstract

Presently, there are two commonly used techniques in the implementation of Viterbi algorithm. One is by a single sequential processor and the other is by many fully parallel processors. In this paper, we consider performing Viterbi decoding based on linear systolic array which is suited for VLSI implementation. For efficient operations under a large number of states, strongly connected trellis decoding is considered. Issues dealing with throughput rate, computations overhead ratio, and path memory updating are also discussed.

I. INTRODUCTION

Viterbi algorithm ([1]-[5]) is well known to be an efficient method for the realization of maximum likelihood decoding of convolutional codes. This algorithm can be thought of as finding the shortest path in the trellis diagram to match the received sequence by a dynamic programming technique. One of the drawbacks of the Viterbi algorithm is that the decoder complexity increases exponentially in proportion to the constraint length K. Since the error probability of convolutional codes also decreases exponentially in K, larger constraint length codes are of great interest for high performance systems. Presently, two ways of implementing Viterbi decoding, use either a single sequential processor ([4],[5]) or many fully parallel processors ([5],[6]). But as the constraint length becomes larger, the throughput rate of a single sequential processor decreases exponentially. Though the fully parallel processors can still have high throughput rate, it suffers from the fact that the topological issues make the design difficult for VLSI implementation. In addition, also as the code structure is changed, it may require a new design procedure. This means that the fully parallel processors have much inflexibility in implementation.

A systolic array ([7]-[10]) is a network of processors that rhythmically process and pass data among themselves. It provides pipelining, parallelism, and simple adjacent neighbor cell interconnection structures so that it is suitable for VLSI implementations. Many basic matrix operations can be performed by systolic array ([8]). In this paper, we shall consider the implementation of a Viterbi decoder by using a systolic array approach. It is shown that as the constraint length becomes larger, the systolic Viterbi decoder still has a very regular structrue as well as a moderate throughput rate gain over a single sequential processor.

II. VITERBI DECODING BY SYSTOLIC ARRAY

In general, the Viterbi decoding process can be divided into three stages: branch metric generation, path metric updating, and path memory (survivor) updating. The branch metric generation consists only of simple computations and the path memory updating basically consists only of memory read and write. Both of these are much easier to handle than the path metric updating. The key issue in the implementation of Viterbi algorithm lies in the way of how the path metric updating is performed. Different kinds of internal path metric updating and organization lead to different kinds of decoder realization. In this paper, we consider a new way to perform path metric updating using a linear systolic array.

There are several ways to consider the Viterbi decoding process. Perhaps the simplest way is to consider it from the point of view as solving the shortest path problem in a weighted graph whose weights are time-varying [3]. The latter problem can be formulated simply as a matrix-vector multiplication problem.

Let us consider a convolutional code of code rate 1/n and constraint length K [5]. Let $N=2^K$ be the number of states.

Let $\mathscr{P}$ be a 1 x N row vector whose j^{th} element is denoted by P_{oj} and represents the accumulated path metric from state o to state j. Let $\mathscr{B}$ be a N x N matrix, whose $i\text{-}j^{th}$ element is denoted by b_{ij}. and represents the branch metric from state i to state j. Then the Viterbi decoding process can be formulated as the repetitions of

$$\mathscr{P} \longleftarrow \mathscr{P} \cdot \mathscr{B} \tag{1}$$

For example

$$P_{oj} \longleftarrow \begin{matrix} P_{oo} \otimes b_{oj} \oplus P_{o1} \otimes b_{1j} \\ \oplus \ldots \oplus P_{oi} \otimes b_{ij} \\ \oplus \,.. \oplus P_{o,N-1} \otimes b_{N-1,j} \end{matrix} \tag{2}$$

where $\otimes$ means conventional additions and $\oplus$ means the operation of taking maximum. Thus, the above expression means

$$P_{oj} \longleftarrow \max \left\{ \begin{matrix} P_{oo} + b_{oj},\ P_{o1} + b_{1j} \\ ,\ldots,\ P_{oi} + b_{ij} \\ ,\ldots,\ P_{o,N-1} + b_{N-1,j} \end{matrix} \right\} \tag{3}$$

or

$$P_{oj} \longleftarrow \max_{0 \le i \le N-1} \left\{ P_{oi} + b_{ij} \right\} \tag{4}$$

Of course (4) is just the updating of the path metric. To determine from which state, say $\hat{i}$, the path of maximum metric (the most likely path) came from, it has the form of

$$\hat{i} \longleftarrow \max^{-1}_{0 \le i \le N-1} \left\{ P_{oi} + b_{ij} \right\}, \tag{5}$$

where $\max^{-1}$ means to take the i that yields the maximum value of $P_{oi} + b_{ij}$. (5) is used to keep track of the most likely path. For any convolutional code, some b_{ij} may not exist, but this problem can be temporarily solved by making b_{ij} be a very large negative number so that this branch will never be chosen.

From the previous discussion, the Viterbi decoding process is formulated in terms of matrix vector operations. However, it is known that systolic array ([8]) can be used efficiently for matrix-vector operations. Specifically, Figure 1 shows the Viterbi decoding process as implemented on a linear systolic array for an 8-state convolutional code, whose trellis diagram is shown in Figure 2. In the systolic array, R_j stays inside the processor while the P_{oj} are moving to the right and the b_{ij} are moving down. All of the movements are synchronized. Each cell has two inputs: P from the left and b from the above. It is capable of performing the addition of P and b followed by updating R to be the maximum of P + b and R. After that, it will transfer its current P to the right cell and obtain a new P from the left cell in addition to a new b from above. It turns out that after eight steps, R_k will successively have the updated path metric P_{ok}. Figure 1 only describes one step of $\mathcal{P} \leftarrow \mathcal{P}\cdot\mathcal{B}$. As the Viterbi decoding process is the repetitions of $\mathcal{P} \longleftarrow \mathcal{P} \cdot \mathcal{B}$, what really happens is that the next set of branch metric is fed right on top of the current set of branch metric. Once a new path metric is obtained, (i.e., R_k contains the new path metric P_{ok}), it is fed into the systolic array from the left so that there is no gap in the whole process.

The systolic array in Figure 1, can be considered as the "P-move and R-stay" scheme due to the characteristic of P and R. Another possibility given by the "P-stay and R-move" scheme, is shown in Figure 3, where b_{ij} is still moving downward but now P stays inside each cell while R is moving to the right. Of course, all the movements are also synchronized. Note that the set of branch metric in Figure 3 is the transposition of that in Figure 1. Now, each cell is doing slightly different operations than previously. It is still computing P+b and picking the maximum of P+b and R. The only difference is that P stays and R moves. Once R_k comes out from the right end cell, it will contain the new path metric P_{ok} and is fed into the k^{th} cell. The new set of branch metric is fed right on top of the current set of branch metric as before.

III. STRONGLY CONNECTED TRELLIS DIAGRAM

In Figure 1 and Figure 3, some branch metrics are squared which means that there exists a non-trivial branch between two specific states in the state (or trellis) diagram. Note that only 2/8th of the branch metrics are squared, which means that at any time instant, only 2/8 of the processors are doing meaningful work. This inefficient situation becomes worse as the number of states increases. This low efficiency is due to the low connectivity in the trellis diagram. If the trellis diagram is strongly connected, (i.e., there exists a non-trivial branch between any two states), each processor will be doing meaningful work and the efficiency increases to 100%.

Now, let us consider Figure 4. Note that in 3 steps, every state is reachable from any other state. If this 3-branch metric is combined to be a single "composite" branch metric, we will have a strongly connected trellis diagram. Then Viterbi algorithm will be working on a strongly connected trellis diagram instead of the previous low connectivity trellis diagram. Of course, how to generate these composite branch metrics needs more consideration. It turns out that for any convolutional code, the original trellis diagram can always be transformed to be strongly connected. The reason is that the connectivity in the trellis diagram has a very high degree of regularity. These results are stated in the following Lemma and Theorem. Their proofs are omitted here.

Lemma:
Consider a convolutional code of code rate 1/n and constraint length K. Let the states be named from 0 to 2^K-1. After renumbering the states in a certain manner, the connectivity

(non-trivial branch) in the trellis diagram permits state i to reach only state ($2i \bmod 2^K$) and state ($2i+1 \bmod 2^K$), where $0 \le i \le 2^K-1$.

Theorem:
For any given convolutional code of code rate 1/n and constraint length K, the smallest number of m needed to obtain a strongly connected trellis diagram from the m-step trellis diagram is K. Furthermore, in the K-step trellis diagram, there is only one path between any two states.

IV. THROUGHPUT RATE AND COMPUTATIONS OVERHEAD RATIO

In this section we consider the throughput rate and computations overhead ratio of the systoliç Viterbi decoder for a convolutional code of code rate 1/n and constraint length K. Let R be the throughput rate in bits/cycle where bits means the number of input bits to the decoder and cycle is the time to finish one addition followed by one comparison. Let G be the throughput rate gain over a single sequential processor. Let COR be the computations overhead ratio defined as the ratio of the number of add/comp in a systolic array over that in a single sequential processor given the same time period. Then we can show that:

$$R = n \times K/2^K \quad \text{bits/cycle}$$
$$G = 2 \times K$$
$$COR = 2^{K-1}/K$$

Example: Code rate 1/2 and constraint length K

K	1	2	3	4	5	6	7
R	2/2	4/4	6/8	8/16	10/32	12/64	14/128
G	2	4	6	8	10	12	14
COR	1	1	4/3	8/4	16/5	32/6	64/7

Note that both G and COR shown above are very similar to that of systolic DFT over FFT. Actually, there are close philosophical similarities between systolic Viterbi decoder and systolic DFT. In both problems, more computations are introduced in comparison to the original problems (i.e., sequential Viterbi decoder and FFT algorithm) so that the problems have more well-behaved structures and can be performed by many pipelining and parallel processors to speed up the throughput rate.

V. PATH MEMORY UPDATING

One of the advantages of the systolic Viterbi decoder over a fully parallel Viterbi decoder is that it does not have difficulty in updating the path memory (survivors). The main reason is that the updated path metric coming out successively instead of coming out almost simultaneously as in a fully parallel Viterbi decoder. This characteristic makes the systolic Viterbi decoder know precisely from which state the path of maximum metric came from and also be able to update all the survivors in the memory successively as the memory can only provide one read/write at a time.

Now, let us consider in more details how the path memory is updated in a systolic Viterbi decoder. As stated before, we assume that the code rate is 1/n and the constraint length is K. Let Pm(j), $0 \le j \le 2^K-1$, be the path memory of state j. Note that it is a bit string. Recall that P_{oi} is the accumulated path metric from state o to state i and b_{ij} is the composite branch metric from state i to state j in a strongly connected trellis diagram. We have

$$P_{oj} \longleftarrow \max_{0 \le i \le 2^K-1} \left\{ P_{oi} + b_{ij} \right\} . \qquad (4)$$

and also

$$\hat{i} \longleftarrow \max^{-1}_{0 \le i \le 2^K-1} \left\{ P_{oi} + b_{ij} \right\} , \qquad (5)$$

Then the path memory of state j is updated as

$$Pm(j) \longleftarrow Pm(\hat{i}) \ \& \ j.$$

where & means bit concatenation and j is in binary representation of state j as used in the Lemma.

VI. CONCLUSIONS

The importance of systolic array implementation in Viterbi decoding is not only that it is well structured for VLSI implementation but also that it has essentially the same structure for any other codes. Many other forms of systolic array for implementing Viterbi decoding are possible. It is also possible to generalize the result considered here to other convolutional codes of different code rates as well as to other applications of Viterbi algorithms in communications. These results will be presented elsewhere.

VII. ACKNOWLEDGEMENT

This work was partially supported by NASA/Ames research contract NAG-2-304.

VIII. REFERENCES

1. A. J. Viterbi, 'Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm', IEEE Trans. Inf. Theory, IT-13 (1967), pp 260-269.

2. J. K. Omura, 'On the Viterbi Decoding Algorithm', IEEE Trans. Inf. Theory, IT-15 (1969), pp 177-179.

3. G. D. Forney,'The Viterbi Algorithm', Proc. IEEE, Vol. 61 (1973), pp 268-278.

4. A. J. Viterbi, and J. K. Omura, Principles of Digital Communication and Coding, McGraw-Hill, 1979.

5. S. Lin and D. J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, 1983.

6. J. B. Cain and R. A. Kriete, 'A VLSI R=1/2, K=7 Viterbi Decoder', IEEE Proc. of the 1984 NAECON, May 1984.

7. H. T. Kung, 'Let's Design Algorithms for VLSI Systems', Proc. of Caltech Conf. on VLSI, 1979.

8. H. T. Kung, 'Why Systolic Architectures?', IEEE Computer Magazine, Jan. 1982.

9. L. J. Guibas, H. T. Kung and C. D. Thompson, 'Direct VLSI Implementation of Combinational Algorithms', Proc. of Caltech Conf. on VLSI, 1979.

10. C. Mead and L. Conway, 'Introduction to VLSI Systems', Chapter 8.3, Addison-Wesley, 1980.

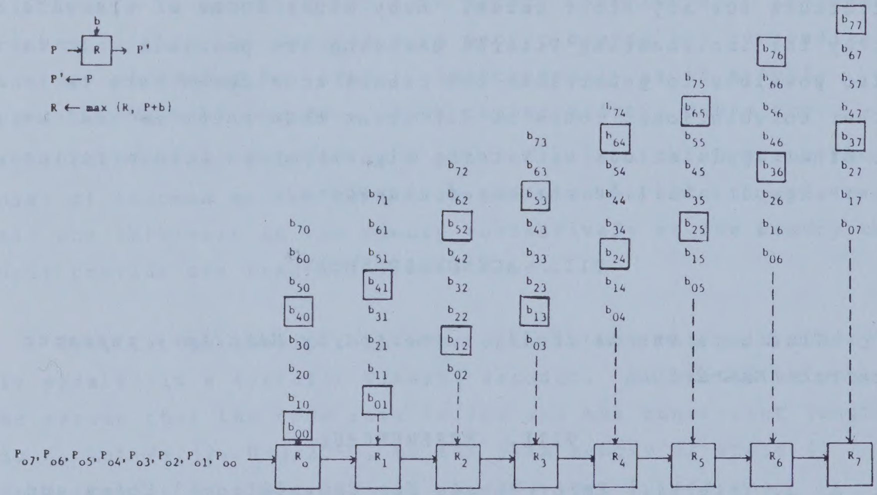

b
P → R → P'
P' ← P
R ← max (R, P+b)
b70 b60 b50 b40 b30 b20 b10 b00
b71 b61 b51 b41 b31 b21 b11 b01
b72 b62 b52 b42 b32 b22 b12 b02
b73 b63 b53 b43 b33 b23 b13 b03
b74 b64 b54 b44 b34 b24 b14 b04
b75 b65 b55 b45 b35 b25 b15 b05
b76 b66 b56 b46 b36 b26 b16 b06
b77 b67 b57 b47 b37 b27 b17 b07
P07, P06, P05, P04, P03, P02, P01, P00
R0
R1
R2
R3
R4
R5
R6
R7

Figure 1. A Systolic Array Implementation of Viterbi Decoding

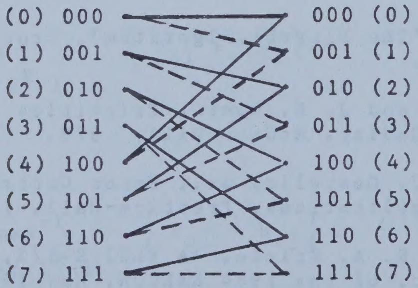

(0) 000
(1) 001
(2) 010
(3) 011
(4) 100
(5) 101
(6) 110
(7) 111
000 (0)
001 (1)
010 (2)
011 (3)
100 (4)
101 (5)
110 (6)
111 (7)

Figure 2. An 8-State Trellis Diagram

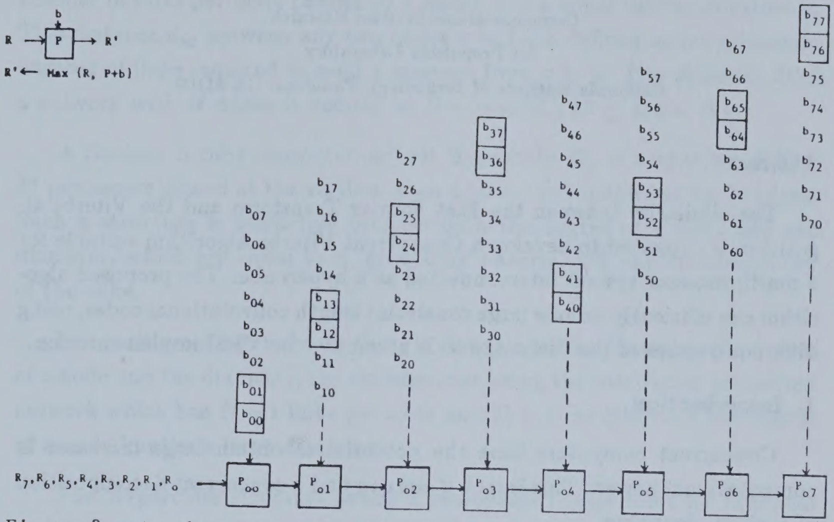

b
R
P
R'
R' ← Max (R, P+b)
b07 b06 b05 b04 b03 b02 b01 b00
b17 b16 b15 b14 b13 b12 b11 b10
b27 b26 b25 b24 b23 b22 b21 b20
b37 b36 b35 b34 b33 b32 b31 b30
b47 b46 b45 b44 b43 b42 b41 b40
b57 b56 b55 b54 b53 b52 b51 b50
b67 b66 b65 b64 b63 b62 b61 b60
b77 b76 b75 b74 b73 b72 b71 b70
R7,R6,R5,R4,R3,R2,R1,Ro
Poo
Po1
Po2
Po3
Po4
Po5
Po6
Po7

Figure 3. Another Systolic Array Implementation of Viterbi Decoding

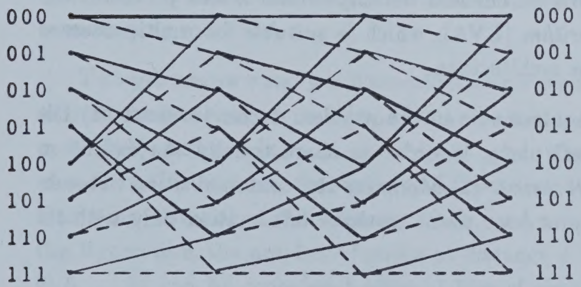

000
001
010
011
100
101
110
111
000
001
010
011
100
101
110
111

Figure 4. A Strongly Connected 8-State Trellis Diagram.

Concurrent Viterbi Algorithm on a Hypercube.

Fabrizio Pollara
Communications Systems Research
Jet Propulsion Laboratory
California Institute of Technology, Pasadena, CA 91109

Abstract:

The similarity between the Fast Fourier Transform and the Viterbi algorithm is exploited to develop a Concurrent Viterbi Algorithm suitable for a multiprocessor system interconnected as a hypercube. The proposed algorithm can efficiently decode large constraint length convolutional codes, using different degrees of parallelism, and is attractive for VLSI implementation.

1. Introduction.

Concurrent computers have the potential to obtain large increases in computational power. This is true if one can find a concurrent decomposition of a given algorithm.

We consider the Viterbi Algorithm for decoding $(m, 1/n_0)$ convolutional codes, where m is the memory (constraint length $= m+1$) and $1/n_0$ is the code rate, and we describe an efficient decomposition of this problem, the *Concurrent Viterbi Algorithm* (CVA), which is suitable for multiprocessor systems with a *Hypercube* architecture.

There are two key requirements in the problem decomposition: (1) Divide the problem in equal parts, in order to share equally the execution time available in each processor. (2) Minimize the communication between the parts, so that each processor needs to share information only with its neighbors.

Fox [1] has shown that the Hypercube is a natural topology for the binary Fast Fourier Transform (FFT), and high efficiency can be obtained with this structure. We will show that there is a simple correspondence between the connectivity of the FFT algorithm and the trellis diagram of the Viterbi Algorithm. Therefore, efficient methods for implementing the Viterbi Algorithm on a Hypercube computer can be developed.

This work was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

2. Hypercube Computer Structure.

In general, it is desirable for an interconnection structure to have a low number of links per node (*degree of a node*), and a small *internode distance.* This distance d_{xy} between any two nodes x and y is defined as the minimum number of links required to send a message from x to y. The *diameter* D of a network with N nodes is defined as $D = max\{d_{xy} \mid 0 \leq x, y < N\}$.

A Boolean n-cube computer, or just Hypercube [2], is a network of $N = 2^n$ processors placed at the vertices of an n-cube, and connected by its edges. Such a structure is attractive because both the degree of a node and the diameter, which are equal to n, grow only *linearly* with the dimensionality of the cube.

In a multiprocessor network there is always a tradeoff between the degree of a node and the diameter, the extreme case being the completely connected network which has $N-1$ links per node and $D = 1$. In general a low degree of a node implies a large diameter.

The Hypercube structure seems a reasonable compromise for practical multiprocessor systems, except for the disadvantage of offering limited input/output capability. This negative aspect, which is less important for computationally intensive (vs. I/O intensive) algorithms, is furthermore mitigated by the small diameter of the network, and the simplicity of efficient broadcasting methods.

Two processors x and y will be called *neighbors* if their binary labels differ only in one position, i.e. $x = [x_{n-1}, x_{n-2}, \ldots, x_k, \ldots, x_0] = \sum_{k=0}^{n-1} 2^k x_k$, where $x_k \in \{0, 1\}$, and $y = [x_{n-1}, x_{n-2}, \ldots, \bar{x}_k, \ldots, x_0]$, where $\bar{x}_k$ is the complement of x_k . In this context, the distance d_{xy} between two processors is just the Hamming distance of their binary labels, and neighbors have $d_{xy} = 1$. In the Hypercube, the number of nodes at distance d from a given node (node $[0, 0, \ldots, 0]$ can be considered without loss of generality) is $N_d = \binom{n}{d}$, so that the *average distance* is $\bar{d} = \frac{1}{N-1} \sum_{d=1}^{n} d\binom{n}{d} = \frac{n}{2}\frac{N}{N-1} \approx \frac{n}{2}$.

3. Equivalence of Networks.

We now define two basic network structures: the m-stage Viterbi algorithm trellis (de Brujin graph), shown in fig. 1 for $M = 2^m = 8$ states, and the FFT decimation-in-time graph of fig. 2.

Let $x = [x_{m-1}, \ldots, x_0]$ be the binary index of a state or node of the graphs, and $u = [u_0, \ldots, u_k, \ldots, u_{m-1}], u_k \in \{0, 1\}$, be a sequence of input bits which define one of the two possible paths out of each node, where k represents the stage.

First, consider the elementary transformation $\sigma(x, u_k)$ which describes the state transitions at stage k, and is defined as

$$\sigma(x, u_k) = \sigma([x_{m-1}, \ldots, x_0], u_k) \triangleq [u_k, x_{m-1}, \ldots, x_1]. \tag{1}$$

This is a *cyclic right shift* of x ,with x_0 replaced by u_k . Then the transformation $\sigma(\cdot, \cdot)$ completely describes the transitions of the graph in fig. 1, and the m-stage transformation from a state x at stage 0 to state y at stage m is defined as

$$y = \sigma^{(m)}(x, u) \triangleq \sigma(\sigma(\ldots\sigma(\sigma(x, u_0), u_1)\ldots), u_{m-1}). \tag{2}$$

The elementary transformation needed to describe fig. 2 is given by

$$\omega(x, u_k) \triangleq [x_{m-1}, \ldots, x_{k+1}, u_k, x_{k-1}, \ldots, x_0], \tag{3}$$

which replaces x_k by u_k . The complete m-stage graph of fig. 2 is then described by the transformation $\omega^{(m)}(x, u)$,

$$y = \omega^{(m)}(x, u) \triangleq \omega(\omega(\ldots\omega(\omega(x, u_0), u_1)\ldots), u_{m-1}) = [u_{m-1}, \ldots, u_0]. \tag{4}$$

Now, it is easy to verify that, at the m^{th} stage,

$$\sigma^{(m)}(x, u) = \omega^{(m)}(x, u).$$

Therefore, the two networks lead from a given state x to the same state y, after m stages.

The equivalence of the two networks can be further expressed, at any stage, by defining the cyclic right shift operator

$$\rho^{(k)}(x) = [x_{k-1}, x_k, \ldots, x_{m-1}, x_0, \ldots, x_{k-2}]$$

$(\rho^{(m)}(x) = \rho^{(0)}(x) = x)$, and verifying that,

$$\sigma^{(k)}(x, u) = \rho^{(k)}(\omega^{(k)}(x, u)). \tag{5}$$

This result shows that, if we relabel each node x of the graph in fig. 1 with the label $\hat{x} = \rho^{(k)}(x)$ at stage k, the two networks are *functionally* and *topologically* equivalent, that is they are just two different ways of drawing the same network. A given path generated by an input sequence u visits the same nodes in fig. 1 and fig. 2, if we relabel each node x in fig. 1 by $\rho^{(k)}(x)$.

Having established this formalism on networks, we can now apply the above results to the study of algorithm structures, in particular to the Viterbi algorithm and its relationship with the radix-2 FFT.

4. Concurrent Viterbi Algorithm.

Consider a multiprocessor system with N processors located at the vertices of an n-cube and linked only by the edges of the cube (see example for n=3 in fig. 3a). Note that the nodes are labeled by a n-bit binary number, so that the i^{th} bit is the coordinate of a node along the i^{th} dimension.

If in fig. 2 we collapse the horizontal dimension, we obtain a graph which is exactly identical to that of fig. 3, i.e. with the same connections between nodes. This observation, as explained in [1] and [3], suggests a natural way to implement the FFT on a Hypercube computer.

The implementation of the FFT on the Hypercube can be stated more formally if we define the Hypercube (m-cube) network by the transformation,

$$\gamma[x_{m-1},\ldots,x_k,\ldots,x_0] = [x_{m-1},\ldots,\bar{x}_k,\ldots,x_0] \tag{6}$$

where $\bar{x}_k$ is the complement of x_k , and observing that the transformation in (3) can always be obtained by (6), since $u_k \in \{0,1\}$ and $u_k \equiv x_k$ requires no communication (self-loop).

To perform the first stage of fig. 2, let the nodes of fig. 3 communicate along the first dimension, and so on for each stage and dimension. In this way, the links provided by the n-cube are just those necessary to perform the FFT. This implementation on the n-cube is possible since the FFT requires communication only between neighboring nodes of the cube.

At first, the network of fig. 1 might seem to require communication between distant nodes on the cube. But this problem can be easily overcome if we relabel the nodes as discussed in sec. 3. Specifically, at stage k, processor x will represent state $\rho^{(k)}(x)$ of the Viterbi trellis. Processor transitions are described by the graph of fig. 2, while state transitions are given by fig. 1, as desired. Therefore a Viterbi decoder can be efficiently implemented on the Hypercube, exactly as for the FFT. The similarity between FFT and

Viterbi trellis was previously pointed out by Forney [3], but apparently not exploited in any practical way.

Each processor, at stage k, receives the accumulated metric and survivor of its neighbor along dimension k of the cube, and performs the usual comparison and update. In a practical Viterbi decoder for a $(m, 1/n_0)$ code, the number of stages required to obtain a performance very close to optimum is approximately $5m$. Notice that, when the stage k is a multiple of m, the state and processor labels are identical, since $\rho^{(m)}(x) = x$, so that we may easily select the decoded bit belonging to the survivor with highest likelihood. However, in order to minimize internode communications, it may be more advantageous to increase slightly the number of stages and read the decoded bit at node zero, which simplifies I/O operations (see sec. 5).

5. Message Broadcasting.

Performing the CVA requires that blocks of data be loaded in every processor: this operation is called *broadcasting.* In the Hypercube [2], data from the host processor is directly exchanged only through node zero (the origin of the cube). Therefore, an efficient concurrent method is required to broadcast a message from node zero to all other nodes. Since the diameter of a n-cube is n, a lower bound on the broadcasting time is n time units (where the unit is the time to send a message to a neighbor).

Assume that message A is in node zero, at time zero. In each subsequent time slot k, send messages in parallel from each node $x = [x_{m-1}, \ldots, x_{k+1}, 0, x_{k-1}, \ldots, x_0]$ to each node $\gamma(x)$, the neighbors along dimension k. After n time units, message A has propagated to all nodes.

Even though this method does not minimize the number of communications (with the advantage of a very simple indexing), it optimizes the total broadcasting time to n time units. The result is clearly optimum, since it achieves the lower bound.

6. Decoders for Large Constraint Length.

Existing Hypercube computers have up to 128 nodes ($n = 7$), and will soon be extended to 1,024 nodes ($n = 10$). Yet, in order to decode powerful convolutional codes with $m > 10$, one needs to obtain algorithms which assign more than one state per processor. This need is dictated not only by the practical limitations on the physical size of the computer, but also by the goal of achieving high computational efficiency.

The *efficiency* η of a parallel computer is defined as,

$$\eta = \frac{N_o t_o}{N_o t_o + N_t t_t} = \frac{\text{sequential alg. time}}{N \times (\text{parallel alg. time})} ,$$

and the *speed-up* σ is given by $\sigma = \eta N$, where N is the number of processors, N_o is the number of parallel operations (butterflies), t_o is the time required by each operation, N_t is the number of parallel data transfers, and t_t is the communication time.

When the number of states M of the decoder is larger than the available number of processors N, states can be grouped in sets of $S = 2^s$ states per processor, where $M = SN$. To see how this is possible for $S = 2$ in the proposed CVA implementation, consider fig. 2 and group each pair of nodes into one processor $P_{\lfloor \frac{i}{2} \rfloor}$, for all nodes i ,$i = 0, \ldots, 7$, where $\lfloor j \rfloor$ is the largest integer less or equal to j. Similarly, with 2 processors $P_{\lfloor \frac{i}{4} \rfloor}$, $i = 0. \ldots, 7$, we obtain the case $S = 4$. The extreme cases, $S = M$ and $S = 1$ represent the completely sequential and completely parallel decoder, respectively. Intermediate cases represent different *degrees of parallelism* or a different *granularity* of the algorithm, which is defined as the amount of computation between successive data transfers. In general, the efficiency increases with the granularity.

During m stages of the CVA, the number of parallel data transfers is,

$$N_t = S(m - s) = \frac{M}{N} n , \tag{7}$$

since only $m - s$ stages of fig. 2 need to communicate with neighbors (s stages do only internal computations). The number of parallel operations (butterflies) N_o is given by :

$$N_o = \frac{S}{2} m = \frac{M}{2N} m. \tag{8}$$

As an example, N_t and N_o are plotted in fig. 4 for $M = 64$. From (7) and (8) we have that the efficiency η decreases as S decreases from N to 1 , and, for $N = M$, η is equal to the ratio $\frac{t_o}{t_o + 2t_t}$, which depends on the hardware implementation.

The CVA with $S > 1$ is useful because it allows the implementation of complex decoders, avoiding the pin-limitation constraint problem encountered in existing VLSI decoders. This problem is due to the particular partitioning of the traditional decoder into a branch metric computation block and a path memory storage block. This partition requires a rapidly increasing number of pins in the VLSI chips. The proposed CVA has instead a number of connections per node increasing only linearly with the dimensionality of the cube, and is therefore more suitable for VLSI implementation.

Furthermore, the CVA has been extended to high rate $(m, k_0/n_0)$ codes with $k_0 > 1$, where k_0 is the number of input bits in the encoder. This extension is possible only for a limited range of m, k_0 and S, the number of states per processor. An example for $(3, 2/n_0)$ codes is given in fig. 5, where $S = 2$.

Given a rate $1/n_0$ code it is known how to generate all the ***punctured*** codes of rate k_0/n_0, $k_0 > 1$. Since this codes involve only pairwise comparisons at each stage, it is certainly possible to decode them with the CVA. It must be noted however that punctured codes require more stages and this is equivalent to linking nodes of the Hypercube which are not neighbors, by using multiple stages.

7. Conclusion.

The proposed CVA decoder has been implemented and successfully tested on a 64-node Hypercube computer for $(m, 1/n_0)$ codes, with $m = 2, \ldots, 6$. Present results confirm the usefulness of the CVA for large constraint length codes.

8. References.

[1] G.C. Fox and S. W. Otto, "Algorithms for concurrent processors", *Physics Today*, May 84, pp. 50-59.

[2] C.L. Seitz, "The Cosmic Cube", *Communications of the ACM*, Vol. 28, No. 1, Jan. 85, pp. 22-33.

[3] M.C. Pease, "An adaptation of the Fast Fourier Transform for parallel processing", *Journal of the ACM*, Vol.15 No. 2, April 68, pp.252-264.

[4] G.D. Forney, "The Viterbi algorithm", *Proc. IEEE*, Vol. 61, No. 3, March 73, pp. 268-278.

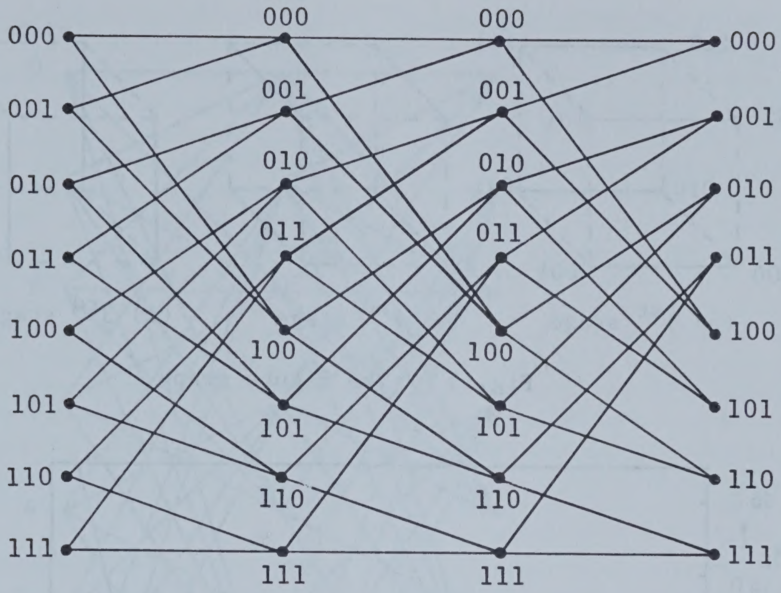

000
001
010
011
100
101
110
111
000
001
010
011
100
101
110
111
000
001
010
011
100
101
110
111
000
001
010
011
100
101
110
111

Fig. 1 - m-stage Viterbi algorithm trellis (m=3)

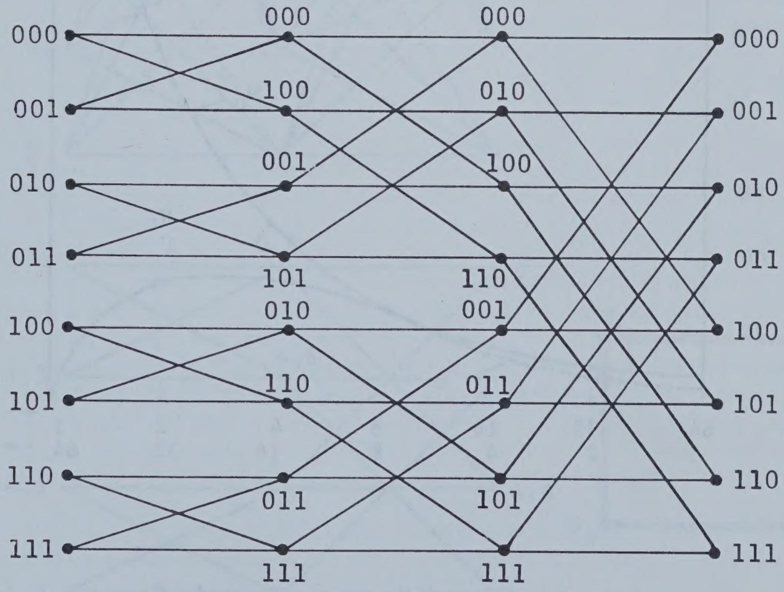
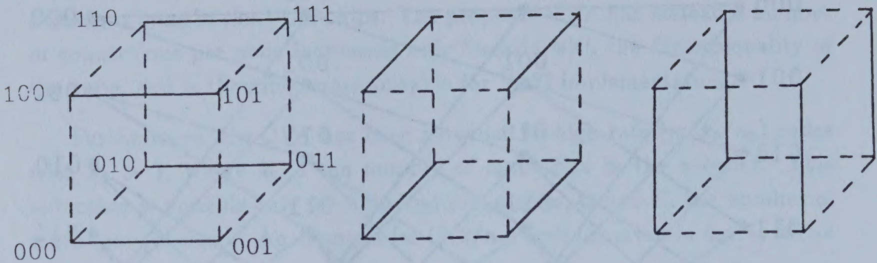

000
001
010
011
100
101
110
111
000
100
001
101
010
110
011
111
000
010
100
110
001
011
101
111
000
001
010
011
100
101
110
111

Fig. 2 - FFT decimation-in-time graph (m=3)


110
111
100
101
010
011
000
001

(a) 1st stage (b) 2nd stage (c) 3rd stage

Fig. 3 - The 3-cube graph.

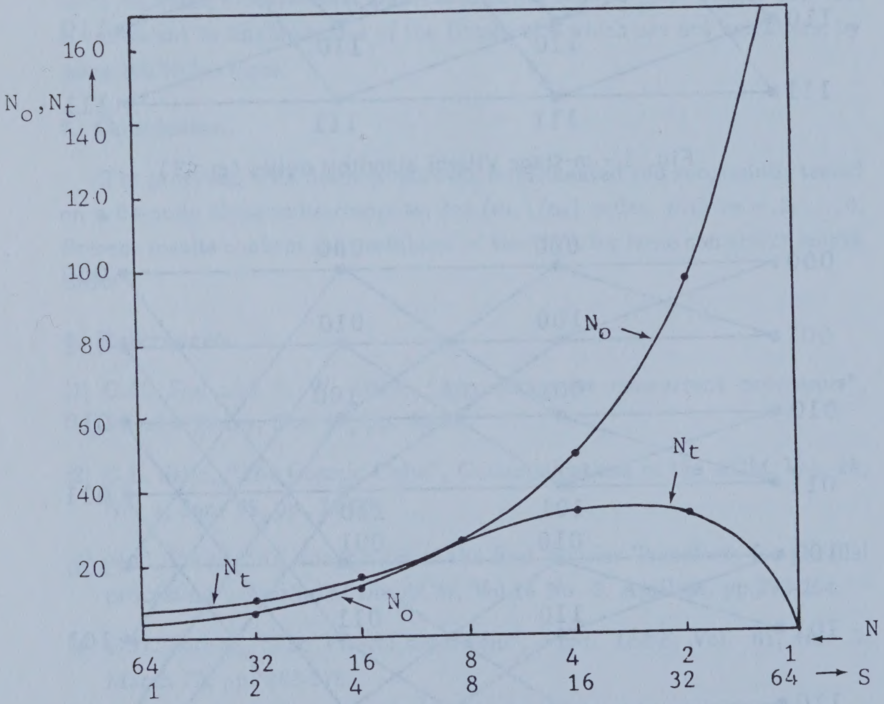


Fig. 4 - N_o and N_t vs. S and N .

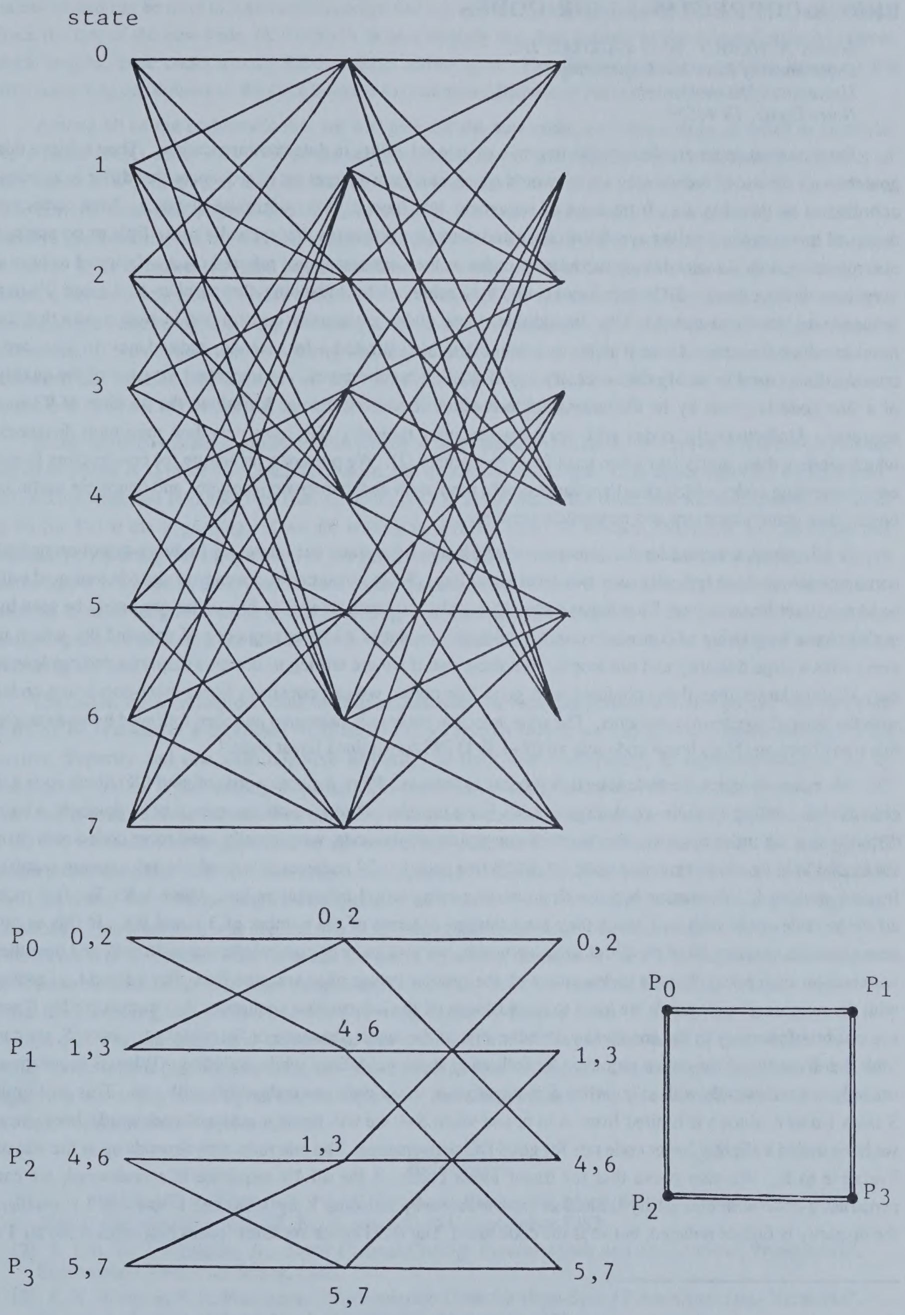

state
0
1
2
3
4
5
6
7
P0
0,2
P1
1,3
P2
4,6
P3
5,7
0,2
4,6
1,3
5,7
0,2
1,3
4,6
5,7
P0
P1
P2
P3

Fig. 5 - Decoder for (3,2/n_0) code, with S=2 .

ERROR-CORRECTING LINE CODES

MARK A. HERRO AND LAIZHAO HU
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

Error control codes are designed to improve system reliability in data communications. They achieve this goal through the use of redundancy -- codewords are chosen from a larger set of sequences that differ in as many coordinates as possible, i.e., from a set of sequences that have a large minimum distance. Line codes are designed to aid in the receiver synchronization and detection processes, but typically make little or no attempt at error-correction. Line codes (or modulation codes as they are sometimes referred to) are designed to have a very low direct current (DC) component which is achieved by balancing the number of **1**'s and **0**'s in a sequence as closely as possible [1]. In addition, these codes are usually constructed in such a way that the number of consecutive **1**'s or **0**'s (the run length) is also limited. In essence, redundancy in line code construction is used to satisfy the necessary signal shaping requirements. An important measure of the quality of a line code is given by its disparity, which is equal to the number of **1**'s minus the number of **0**'s in a sequence. Unfortunately, codes with very low disparity typically also have very low minimum distances, which renders them inefficient when used for error control [2]. We present a technique for constructing linear error-correcting codes which simultaneously satisfy run length and disparity constraints, and hence are useful in optical data communications and magnetic recording.

In this paper, we consider the construction of binary codes since our interest is in direct-detection optical communication which typically uses two level signalling. Since linear codes are easy to decode, our goal will be to construct linear codes. That linear codes do not always function well as line codes can easily be seen by noting that a long string of consecutive information **0**'s produces a longer sequence of encoded **0**'s, which in turn yields a large disparity and run length. We show that if we are willing to accept a disparity and run length only slightly larger than those obtained with good line codes, we can construct linear error-correcting codes with the desired spectrum constraints. The code selection process is presented and then followed by an example that transforms an (N,K) linear code into an (N+1,K-1) DC constrained linear code.

We can summarize the code selection process as follows. First, the codewords of an (N,K) block code are classified according to their word disparity, s. For example, w^s represents the group of codewords whose disparity is s. In order to satisfy the disparity constraints on the code, we typically need more codewords than are available in an error-correcting code. (A 4B5B line code has 32 codewords but only 16 information words). Instead of using K information bits, we first consider using only I information bits, where $I<K$. We first pick all of the codewords with $s=0$, since they are balanced in terms of the number of **1**'s and **0**'s. If this is not enough words to cover all of the 2^I information words, we pick pairs of codewords, one with $s>0$, and the other with $s<0$, in such a way that the codewords with the smaller values of $|s|$ are used first. For a fixed I, $|s|$ grows with the number of codewords we have to use to cover all the information sequences, but $n=\max|s|\leq N$. If we use one bit of memory in the encoder to store the sign of the whole sequence or "running" disparity, S, we can limit the disparity of the entire sequence by following these guidelines while encoding. When S is positive, we only use codewords with $s\leq0$; when S is negative or 0, we only use codewords with $s\geq0$. This will limit S from 1-n to n, since s is limited from -n to n, and when $S=0$ we will never use an $s<0$ codeword. In essence, we have traded a slightly lower code rate for good DC performance. The rate reduction depends upon the size of I relative to K. We also prove that for linear block codes, if the all 1's sequence is a codeword, we can construct a code with a disparity limited as shown above by reducing K by 1 bit, i.e., I=K-1. (If I is smaller, the disparity is further reduced, but so is the code rate.) The BCH codes are linear codes that contain the all **1**'s

---

This work was supported by NASA Grant NCA2-1R562.

codeword and can be used to construct line codes that can correct the same number of errors as the original code. Since the rate of the new code, (K-1)/(N+1), is only slightly less than the rate of the original code for typical block lengths, these codes usually have positive coding gain. The above encoding algorithm preserves the error correcting capabilities of the code because the minimum distance of the code is not reduced.

Among all of the codewords that we will pick for the new code, we want n to be as small as possible. For a linear (N,K) code, n=N. When the weight distribution of the original code is known, the number of codewords in w^s, for all s, is also known [2], and it is possible to find the minimum value of n for the new code derived from that original code. Unfortunately, the derived code will not, in general, be linear. To preserve the linearity, we have to use a slightly larger n. We do this by first recalling the following distance property. If we complement some bits at the same coordinates in all of the codewords, we do not change the distance between any pair of codewords, but we can change the maximum disparity level, n. For example, among all the codewords in a (7,4) Hamming code, n=N=7, because the all **0**'s and the all **1**'s sequences are codewords. If we complement all 3 parity bits, the minimum distance of the code remains at 3, but now each codeword has at least two **0**'s and two **1**'s. This reduces n from 7 to 3. Of course, this value of n is not guaranteed to be the minimum that can be obtained for a block code of length 7.

Some applications also require the run length to be as small as possible. We prove that for (N,K) codes, whose disparity is at most n for each codeword, the maximum run length, L, is upper bounded by $L \leq N+n-1$. This occurs when the first codeword has all its **0**'s (or **1**'s) at the end of the word, and the next codeword has all its **0**'s (or **1**'s) at the beginning. If we are willing to add one more bit to each codeword, we can break this continuity by putting the complement of the last bit at the end of each codeword. In this way, the run length can be upper bounded by $L \leq (N+n)/2$. In a method similar to that used to decrease the disparity, we can also decrease the run length of some codes by noting that the minimum distance of the code is not changed by reordering the bit sequence for all the words. More research needs to be done to find an optimal strategy for reordering the bits.

The following encoding/decoding algorithm illustrates the concepts presented in this paper. We start with an (N,K) BCH code (or a shortened BCH code). If we restrict I=K-1, we can use two codewords (one with positive disparity and one with negative disparity, or both with 0 disparity) to represent each of the 2^I information words. If the running disparity, S, is positive, the codeword with $s \leq 0$ is used; otherwise the $s \geq 0$ codeword is used. Before transmitting, the parity bits in each codeword are complemented, and the complement of the last codeword bit is appended to the end of the codeword. The decoder first strips the last bit off, complements the parity bits, and then passes the sequence through a BCH decoder. If the first information bit is a 1, the rest of the word is complemented; otherwise the rest of the word is left unchanged, and the decoding is finished. This algorithm was applied to a (13,9) shortened Hamming code (shortened from a (15,11) code), to construct a (14,8) code. This code has the same minimum distance as the Hamming code, and thus can correct one bit error or no more than three erasures per block with the standard Hamming decoder. However, unlike the Hamming code, this code has a running disparity limited from -8 to 10, and its maximum run length is 10. The run length and disparity of this code is a little bit larger than those of typical line codes of similar block length [3], but this code can also correct errors. Although this example considers a rate 4/7 single error-correcting code, higher rate multiple error correcting codes of this type can also be constructed and may be of value in noisier environments.

References

[1] A. X. Widmer, P. A. Franaszek, "A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code", *IBM J. Res. Develop.* Vol. 27, No. 5, pp. 440-451, September 1983.

[2] S. Lin, D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[3] A. X. Widmer, P. A. Franaszek, "Transmission Code for High-Speed Fibre-Optic Data Networks", *Electronic Letters*, Vol. 19, No. 6, pp. 202-203, March 17, 1983.

METHODS FOR PROTECTION OF COMPUTER MEMORIES

M. BLAUM, R. M. F. GOODMAN, and R. J. McELIECE
Electrical Engineering Department
California Institute of Technology, Pasadena, California

ABSTRACT

In this paper we discuss two methods for increasing the lifetime of a computer memory with multiple error-correction-coding (ECC): adding s rows of spare chips with single error correction, versus double error correction. Comparing the two methods in terms of the number of redundant chips, we conclude that sparing is better when the number of rows of chips is large enough. However, if complexity of implementation is considered, 2-ECC may be superior.

1. INTRODUCTION

A computer memory is an M by n array of by-one memory chips, such that the first k columns are information chips and the last $n-k$ columns are redundant chips. Thus k is equal to the width of the computer word. The redundant chips are used to encode and protect the information in general through a single-error-correcting double-error-detecting (SEC–DED) code [2]. Internally, each chip is an ℓ by ℓ array of cells, each of which stores a 0 or a 1. Chip failures can be hard or soft, and we assume that the soft errors which generally only affect a single cell can be easily removed by periodic scrubbing. In the case of hard errors, several types of chip failures can occur: single-cell failures, row failures, column failures, and catastrophic chip failures. Regardless of how bad a chip failure is, the code will correct it whenever no more than one chip in a row is affected. However, it is expected that after a certain amount of time, an uncorrectable pattern will have occurred. In that case, we say that a memory failure has occurred.

A natural problem is to find the mean time between failures (MTBF) in a memory with a SEC–DED code, *i.e.*, how long we have to wait on the average, until an uncorrectable pattern occurs. This problem has been extensively studied ([1]–[3]). These studies show that if we define the coding gain (CG) as the ratio of MTBF with SEC–DED to the MTBF without SEC–DED, then this CG may be as large as several thousands. In this paper we are concerned with methods of producing even higher reliability memory systems. Two methods are of interest, adding spare chips to a SEC–DED code and performing spares switching, or alternatively implementing double error correction (DEC–TED).

2. SPARES SWITCHING

One way of further extending the lifetime of the computer memory is by adding s rows of spare chips to the M by n array which also has 1–ECC. The spare chips act in such a way that each time a chip failure occurs, a connection to a spare chip in the corresponding column is made. In practice, this means that the chip is replaced by one of the s spare chips in the corresponding column. Hence, s failures per column are tolerated before errors are actually corrected continuously.

Call $(CG)_s$ the coding gain when s rows of spare chips are added in this way. We have found the following upper and lower bounds for $(CG)_s$:

$$(CG)_0 + \frac{k}{n} B_2(n, s+1) \leq (CG)_s \leq (CG)_0 + ks \tag{1}$$

where $B_2(n, s+1)$ is the average number of balls we have to place in n cells, until either two cells have $s+1$ balls each or one cell has $s+2$ balls. An expression for $B_2(n, s+1)$ is given by ([1]). Notice that $(CG)_0$ is the coding gain when 1–ECC is implemented with no spares.

3. DOUBLE ERROR CORRECTION

An alternative way of further extending the lifetime of the computer memory is to implement 2–ECC, that is, adding more parity check chips so that we have a double-error-correcting triple-error-detecting (DEC–TEC) code.

Let us call $(CG)'_0$ the coding gain for the memory with a DEC–TEC code. The memory protected by a DEC–TED code can be considered to be an M by n' array of chips, where n' is the number of columns needed to implement the DEC–TED code. If we assume that only cell row and column failures occur, and that they occur with the same frequency, we obtain the following results:

$$(CG)'_0 = \frac{k}{n'} \int_0^\infty e^{-M\ell x}(1 + x + \frac{x^2}{4})^M \, \ell dx \tag{2}$$

and as $M \to \infty$,

$$\frac{(CG)'_0}{(CG)_0} \sim \frac{n}{n'} \sqrt{\ell} \tag{3}$$

For example, if $\ell = 256$ (64K chips), $k = 32$, $n = 39$, $n' = 39$, $n' = 45$, and M a large number, the increase in coding gain when DEC–TED is implemented with respect to the memory with SEC–DED is roughly 14 times.

4. REDUNDANCY COMPARISON

Let us now compare the redundancy of the two methods. Assume that M is large enough so that our asymptotic approximations hold. We wish to estimate the number of spare chips that we have to add such that $(CG)_s \geq (CG)'_0$. If this is less than the number needed for 2-ECC, then sparing is better. For the coding gain condition to hold, we have from Equation (1):

$$(CG)_0 + \frac{k}{n} B_2(n, s+1) \geq (CG)'_0 \tag{4}$$

Replacing (3) in (4) and given that ([1]):

$$(CG)_0 \sim \frac{k}{n} \sqrt{\pi}\sqrt{M} \tag{5}$$

we have the following inequality on s:

$$\left(\frac{n}{n'}\sqrt{\ell} - 1\right) \sqrt{\pi}\sqrt{M} \leq B_2(n, s+1) \tag{6}$$

Using our typical example with $k = 32$, $n = 39$, $n' = 45$, and $\ell = 256$, we obtain:

$$(22.8)\sqrt{M} \leq B_2(39, s+1) \tag{7}$$

Using Equation (7), and performing numerical integration to find $B_2(39, s+1)$, we can find the minimum number of spare rows s that satisfies the inequality for fixed M. Let us call this value $s(M)$. Now, if we add $s(M)$ spare rows, we are adding $39s(M)$ chips, while if we implement a DEC–TED code, we have to add $6M$ chips. Thus if $39s(M) < 6M$ or $s(M) < 2/13M$, we conclude that sparing is better.

The table below compares these values. It can be seen that $s(M)$ grows very slowly. For M around 50, the result is inconclusive, but for $M > 60$ sparing is clearly better.

M	$s(M)$	$2/13M$
50	8	7.7
60	9	9.2
80	10	12.3
100	11	15.4
200	14	30.8
400	19	61.5

However, spares switching is much more complex to implement, and so 2–ECC may turn out to be superior.

References

(1) M. Blaum, "Ph.D. Thesis," Caltech, 1984.

(2) R. M. F. Goodman and R. J. McEliece, "Lifetime analyses of error-control coded semiconductor RAM systems," Proc. IEEE, v. 129, part E (32), 81-85 (May 1982).

(3) R. M. F. Goodman and R. J. McEliece, "Hamming codes, computer memories, and the birthday surprise," Proc. 20th Allerton Conference on Communication, Control and Computering, 672-679 (1982).

THE STRUCTURE AND COMPLEXITY OF LINEAR SUM CODES

TOM FUJA and CHRIS HEEGARD
School of Electrical Engineering
Cornell University
Ithaca, NY 14853

ROD GOODMAN
Department of Electrical Engineering
California Institute of Technology
Pasadena, CA 91125

ABSTRACT

With the advent of large semiconductor random access memories (RAM's), the use of on-chip error control is becoming a necessity. As memories become more dense, soft (or non-permanent) error sources such as alpha particle radiation [1] and circuit noise can dominate RAM operation and make them useless. In addition, recent results [2] have shown that the traditional means of controlling hard (or permanent) defects discovered during the manufacturing process - row/column spare switching - may not be effective for very large memory arrays. Thus, the need for powerful, easily implemented codes for on-chip operation may well become great within a few generations of memory devices.

Two categories of codes designs have surfaced to meet this need. The first is a bi-directional parity check code [3-5] which uses a simple product-code-type design to provide single error correction. The other is a shortened Hamming code implemented on a 256K RAM by Micron Technologies of Boise, Idaho [6].

Linear sum codes (LSC's) are a general class of codes which are suitable for on-chip error correction [7]. They are a generalization of the bidirectional parity check code which allows for multiple error correction. In this paper we will examine the rates which can be achieved and the complexity involved in implementing binary linear sum codes.

I. AN OVERVIEW OF LINEAR SUM CODES

In this section we define what a linear sum code is and show why such codes are suitable for on-chip error protection. We briefly describe some important characteristics of these codes.

(1.1) *Code Definition*

Consider a $k_2 \times k_1$ array of q-ary memory cells containing an arbitrary pattern of symbols. (For most cases of interest, $q=2^b$ for some positive b.) Call this the *information array*. To each of the k_2 rows add $r_1 = n_1 - k_1$ additional symbols such that each row constitutes a codeword from a systematic (n_1, k_1) linear code. Similarly, add to each of the k_1 columns $r_2 = n_2 - k_2$ parity symbols such that each column forms a codeword from a (possibly different) systematic (n_2, k_2) linear code. We call all possible configurations thus constructed for the given constituent codes an (n_1, k_1, n_2, k_2) linear sum code over GF(q).

As described, such codes are similar to product codes, the exception being that the "parities of parities" which are on product codes are not found here. In addition, we include in our definition of linear sum codes a constraint on the decoding of such codes; specifically, we require that the estimate of any symbol in the information array be a function of only the row and column to which that symbol belongs rather than the whole codeword.

The decoding constraint is imposed to make these codes structurally suitable for implementation in semiconductor RAM's. These memory chips are arranged in rectangular blocks, and the addressed bit is specified as a word-line/bit-line intersection. Conceivably, linear sum codewords could be implemented on the word-lines of a RAM; that is, the bit-line address could be split into two smaller addresses, forming a two-dimensional sum codeword in each word line. The decoding constraint means that to decode any given bit *only* those bits sharing one of these smaller addresses with the desired bit are ever considered; thus, the decoding scheme takes advantage of the addressing scheme.

This work has been supported by NSF grant #ECS-835220, NATO grant #215/84, by a grant from AT&T Information Systems, and by the AT&T Bell Laboratories Ph.D. Scholarship program.

(1.2) *Characteristics of Linear Sum Codes*

(1.2.1) *Minimum Distance Properties:* The minimum distance of a linear sum code is not always immediately evident; however, we *can* bound the minimum distance, and we can describe it exactly for a broad class of linear sum codes.

A systematic (n,k) code can, by definition, be split up into an information segment of length k and a redundancy segment of length n-k. In our definition of linear sum codes we construct the information array from the information segments of the constituent codewords; this information array is fundamentally different from the rest of the codeword in that each symbol in the information array is a part of two constituent codewords, whereas the symbols in the rest of the codeword are a part of only one. Thus, it should not be surprising that the minimum distance of a linear sum code depends on the Hamming weight of the information segment of its constituent codes.

Definition: A systematic (n,k) linear code with minimum distance $d_{\min}$ is said to be *unary-minimal* if there exists a codeword of Hamming weight $d_{\min}$ which contains exactly one non-zero symbol in its information segment.

The following lemma describes exactly the minimum distance properties of linear sum codes constructed from unary-minimal.

Lemma 1.1: Consider an (n_1, k_1, n_2, k_2) linear sum code with minimum distance $d_{\min}$. Let $d^1_{\min}$ and $d^2_{\min}$ be the minimum distances of the constituent codes. Then

$$d_{\min} \geqslant d^1_{\min} + d^2_{\min} - 1.$$

Furthermore, equality holds if both the row code and the column code are unary-minimal.

Most codes of interest *do* have the unary-minimal property. Certainly the systematic Hamming codes do, and if sum codes on RAM's are going to proceed beyond the simple bi-directional parity check code, Hamming codes will be among the first constituent codes tried.

(1.2.2) *Error Tolerating Capabilities of Linear Sum Codes:* The decoding constraint which we've put on linear sum codes suggests that minimum distance may not be the best measure of a code's performance; minimum distance gives the number of symbols where two codewords differ, but the decoding constraint tells us that the entire codeword is not available to the decoder.

A better parameter is described as follows. Let b_{ij} be the maximum number of errors which can be *tolerated* in the i^{th} row and j^{th} column when decoding the $(i,j)^{th}$ symbol in the information array. (Note we say tolerated as opposed to corrected; we only want to correct the $(i,j)^{th}$ entry.) If we let

$$b = \min_{i \leqslant k_1,\ j \leqslant k_2} b_{ij}$$

then b indicates how many errors can be tolerated in any row/column combination. We will call such a code a *b error-tolerating linear sum code.*

The following lemma relates the error tolerating performance of a linear sum code with the minimum distances of its constituent codes.

Lemma 1.2: Consider an (n_1, k_1, n_2, k_2) linear sum code over GF(q). Let $d^1_{\min}$ and $d^2_{\min}$ be the minimum distances of the constituent codes. Such a code is capable of tolerating $b = \left\lfloor \frac{d_t - 1}{2} \right\rfloor$ or fewer errors, where d_t, called the *tolerating distance,* is given by

$$d_t = d^1_{\min} + d^2_{\min} - 1.$$

Consider, then, the class of linear sum codes whose constituent codes are unary-minimal. This result tells us that, to decode any symbol in the information array of such a code, we can do no better looking at the entire codeword than we can just looking at the row/column to which the desired symbol belongs.

(1.2.3) *Decoding a Linear Sum Codeword:* As indicated in section (1.1), the proposed use of linear sum codes is in the word-lines of a RAM. During operation, such word-lines are usually changed in a read-modify-write cycle. Thus, in decoding a linear sum codeword, we are concerned solely with estimating symbols *in the information array;* the parity symbols will be re-generated at the end of each cycle and thus need not be estimated.

So consider the problem of estimating a symbol w in the information array based only on the row and column to which w belongs. For a given choice of constituent codes, there is a set of all permissible i^{th}-row/j^{th}-column combinations; it's obvious that this set forms a group under addition. Furthermore, the set of all such combinations which contain a *zero* at the i^{th}-row/j^{th}-column intersection is a subgroup. Our general decoding strategy is as follows: Map the received i^{th}-row/j^{th}-column into the closest permissible combination. This row/column combination lies in a coset of the subgroup described above; make the estimate of the $(i,j)^{th}$ symbol correspond to that coset. This decoding method achieves the b-error-tolerating capability of Lemma 1.2.

II. TWO-ERROR-TOLERATING BINARY LINEAR SUM CODES

From the results presented in section I, we know that to design a two-error-tolerating linear sum code we must use constituent codes which satisfy

$$d_{\min}^{1} + d_{\min}^{2} \geqslant 6, \tag{2.1}$$

where $d_{\min}^{1}$ and $d_{\min}^{2}$ are the minimum distances of the constituent codes. There are two obvious alternatives which satisfy (2.1) with equality:

(1) Let both constituent codes be single-error correcting (SEC) Hamming codes ($d_{\min} = 3$). We will refer to a code so constructed as a SEC/SEC linear sum code.

(2) Let one of the constituent codes be a single-error-correcting/double-error-detecting (SEC-DED) extended Hamming code ($d_{\min} = 4$) and let the other be a single-error-detecting (SED) parity-check ($d_{\min} = 2$). We will refer to such a code as a SEC-DED/DED linear sum code.

In this section we will analyze these two alternatives for the case q=2 (i.e. - binary codes). Specifically, we will demonstrate easily implemented decoding algorithms which attain the two-error-tolerating goal, and we will determine what code rates are achievable for a fixed information array size.

(2.1) *Decoding Algorithms*

The algorithm described in section (1.2.3) is a general means of decoding a symbol in the information array. In this section we will describe specific, easily implemented algorithms for decoding a bit in the information array of a two-error tolerating binary linear sum code of each of the two types.

(2.1.1) *Dual Hamming Code Design:* Suppose the code under consideration is a SEC/SEC code. We wish to estimate the $(i,j)^{th}$ bit in the information array based on the i^{th} row and the j^{th} column.

Let S_r be the syndrome of the i^{th} row, and let S_c be the syndrome of the j^{th} column. For simplicity's sake, let's assume that the syndromes are computed such that they "point" to the estimated error position; that is, $S_r = m$ for $1 \leqslant m \leqslant n_1$ implies that by inverting the m^{th} bit in the i^{th} row, the i^{th} row constitutes a valid codeword from the row code; of course, $S_r = 0$ implies that the i^{th} row already is a valid codeword. Similarly define S_c.

Assume that w was written into the $(i,j)^{th}$ position in the information array and that at a later time we read back a possibly corrupted version y. Let $\hat{w}$ be our estimate of w; then $\hat{w} = 1 + y$ if both S_r and S_c are non-zero and either $S_r = j$ or $S_c = i$; otherwise, $\hat{w} = y$.

(2.1.2) *Extended Hamming Code/Parity Check Design:* Suppose the code under consideration is a SEC-DED/SED binary code. Again we want to estimate the $(i,j)^{th}$ bit in the information array based on the i^{th} row and the j^{th} column.

Assume that the SEC-DED code is on the rows of the sum codeword; define S_r as the syndrome of the i^{th} row. Just as in the last section, choose the S_r such it "points" to a single error; in addition, if a double error is detected, then set $S_r = -1$. Define S_c as the modulo-two sum of the j^{th} column; thus, a single error anywhere in the j^{th} column will yield $S_c = 1$.

Then $\hat{w} = 1 + y$ if either $S_r = j$ or ($S_r = -1$ and $S_c = 1$); otherwise, $\hat{w} = y$.

(2.2) *Achievable Code Rates*

In this section we will compare the two-error tolerating binary codes in terms of optimal rate for a fixed information array size.

In our analysis we assume that the constituent codes are an (n_1, k_1) code and an (n_2, k_2) code and that $k_1 = 2^{l_1}$ and $k_2 = 2^{l_2}$ for integer values $l_1, l_2 \geqslant 0$. Thus, there are 2^L bits in the information array, where $L = l_1 + l_2$.

We will analyze both SEC/SEC and SEC-DED/SED codes; define $Q_A(l_1, l_2)$ to be the number of redundant cells in an (n_1, k_1, n_2, k_2) type "A" linear sum code when $k_1 = 2^{l_1}$ and $k_2 = 2^{l_2}$. (In this section, "A" will be either SEC/SEC - abbreviated S/S - or SEC-DED/SED - abbreviated S-D/S. Similarly define $R_A(l_1, l_2)$ to be the rate of such a code; thus,

$$R_A(l_1, l_2) = \frac{2^{l_1 + l_2}}{2^{l_1 + l_2} + Q_A(l_1, l_2)}.$$

Finally, define $R_A^*(L)$ to be the highest rate for an (n_1, k_1, n_2, k_2) "type A" linear sum code when $k_1 = 2^{l_1}$, $k_2 = 2^{l_2}$, and $L = l_1 + l_2$ is fixed; i.e.,

$$R_A^*(L) \equiv \max\{R_A(l_1, l_2) : l_1 + l_2 = L\}.$$

(2.2.1) *Dual Hamming Code Design:* First consider the case where both the row code and the column code are SEC binary Hamming codes. It can be shown that for $L \geqslant 4$ the optimal rate is achieved by making the information array as square as possible. We state this more precisely in the following lemma:

Lemma 2.1: For $L \geqslant 4$,

$$R_{S/S}{}^*(L) = R_{S/S}\left(\left\lfloor \frac{L}{2} \right\rfloor, \left\lfloor \frac{L+1}{2} \right\rfloor\right) = \begin{cases} \dfrac{2^L}{2^L + (\frac{L}{2} + 1)2^{\frac{L}{2}+1}} & \text{for } even\ L \\ \dfrac{2^L}{2^L + \left\lceil \frac{3L+5}{2} \right\rceil 2^{\frac{L-1}{2}}} & \text{for } odd\ L \end{cases}.$$

To prove this, we will make use of the following lemma, offered without proof:

Lemma 2.2: For fixed $L \geqslant 4$, consider the function $M_L : R \rightarrow R$ defined by

$$M_L(x) = 2^{\frac{L}{2}-x}\left(\frac{L}{2} + x + 1\right) + 2^{\frac{L}{2}+x}\left(\frac{L}{2} - x + 1\right).$$

Then $M_L(x) = M_L(-x)$ and $M_L(x+1) \geqslant M(x)$ for $x \epsilon [0, \frac{L}{2} - 1]$.

To see why this lemma is important, note that for $l \geqslant 2$, it takes $l + 1$ additional bits to provide single error correction to 2^l information bits. (For $l = 2$, this is just the (7,4) Hamming code; for $l \geqslant 3$ it's a shortened version of the standard $(2^{l+1} - 1, 2^{l+1} - (l+1) - 1)$ Hamming code.) Thus, for $l_1, l_2 \geqslant 2$

$$Q_{S/S}(l_1, l_2) = Q_{S/S}(l_2, l_1) = 2^{l_1}(l_2 + 1) + 2^{l_2}(l_1 + 1),$$

and so

$$Q_{S/S}(l_1, l_2) = M_L\left(\frac{|l_1 - l_2|}{2}\right)$$

when $l_1 + l_2 = L$.

So, lemma 2.2 tells us that for even $L \geqslant 4$,

$$\min_{2 \leqslant l_1 \leqslant L-2} Q_{S/S}(l_1, l_2) = M_L(0) = \left(\frac{L}{2}+1\right) 2^{\frac{L}{2}+1}.$$

and for odd L⩾5,

$$\min_{2 \leqslant l_1 \leqslant L-2} Q_{S/S}(l_1, l_2) = M_L\left(\frac{1}{2}\right) = \left\lfloor \frac{3L+5}{2} \right\rfloor 2^{\frac{L-1}{2}}.$$

To find $Q_{S/S}(0, L)$ and $Q_{S/S}(1, L-1)$ we note that the relevant codes are the (3, 1) repetition code and the (5,2) shortened Hamming code. Thus for L⩾3, $Q_{S/S}(0, L) = 2^{L+1} + L + 1$ and $Q_{S/S}(1, L-1) = 3(2^{L-1}) + 2L$; both of these are greater than $M_L(0)$ for L even, L⩾4, and both are greater than $M_L(\frac{1}{2})$ for L odd, L⩾5. Thus Lemma 2.1 is proven.

(2.2.2) *Extended Binary Hamming Code/Parity Check Design:* Now consider the case where the row code is an extended binary Hamming code and the column code is a simple parity check.

It's easy to verify that $Q_{S-D/S}(0, l_2) = 3(2^{l_2}) + 1$, $Q_{S-D/S}(1, l_2) = 2^{l_2+2} + 2$, and $Q_{S-D/S}(l_1, l_2) = 2^{l_1} + 2^{l_2}(l_1 + 2)$ for $l_1 \geqslant 2$. Our goal is to choose l_1 such that $Q_{S-D/S}(l_1, l_2)$ is minimized for a fixed L.

To do this we must first characterize L thusly: for every positive integer L there exist a unique integer $m = m(L) \geqslant 1$ and a unique integer $j = j(L)$, $0 \leqslant j \leqslant 2^m - 2$, such that $L = 2^m - m + j$. We make use of this fact in the following lemma:

Lemma 2.3: For L⩾2, let m=m(L) and j=j(L), as defined above; then

$$R_{S-D/S}{}^* = R_{S-D/S}(l_1^*, l_2^*),$$

where

$$l_1^* = 2^{m-1} + \left\lfloor \frac{j}{2} \right\rfloor \text{ and } l_2^* = 2^{m-1} - m + \left\lfloor \frac{j+1}{2} \right\rfloor.$$

To prove Lemma 2.3, we will make use of the following lemma, given without proof:

Lemma 2.4: For fixed L⩾2, consider the function $N_L : R \rightarrow R$ defined by

$$N_L(x) = 2^{2^{m-1} + \frac{j}{2} - x} + 2^{2^{m-1} + \frac{j}{2} - m + x} \left(2^{m-1} + \frac{j}{2} - x + 2\right).$$

Then $N_L(x-1) \geqslant N_L(x)$ for all $x \in [m - 2^{m-1} - \frac{j}{2} + 1, 0]$, and $N_L(x+1) \geqslant N_L(x)$ for all $x \in [0, 2^{m-1} + \frac{j}{2} - 1]$.

To see the relevance of Lemma 2.4, note that when $L = l_1 + l_2$,

$$N_L\left(m - 2^{m-1} - \frac{j}{2} + k\right) = Q_{S-D/S}(L-k, k) \quad \text{for } k = 0, 1, 2, \cdots, L-2.$$

Thus, the lemma says that for even j,

$$Q_{S-D/S}\left(2^{m-1} + \frac{j}{2}, 2^{m-1} - m + \frac{j}{2}\right) = N_L(0) = \min\{Q_{S-D/S}(l_1, L-l_1) : 2 \leqslant l_1 \leqslant L\} \quad (2.2)$$

For odd j, the lemma says that

$$\min\left\{N_L\left(-\frac{1}{2}\right), N_L\left(\frac{1}{2}\right)\right\} \leqslant Q_{S-D/S}(l_1, L-l_1) \text{ for } l_1 = 2, 3, \cdots, L.$$

Simple analysis shows that $N_L(\frac{1}{2}) \leqslant N_L(-\frac{1}{2})$ and so

$$Q_{S-D/S}\left(2^{m-1} + \frac{j-1}{2}, 2^{m-1} - m + \frac{j+1}{2}\right) = N_L\left(\frac{1}{2}\right)$$

$$= \min\{Q_{S-D/S}(l_1, L-l_1) : 2 \leqslant l_1 \leqslant L\}. \qquad (2.3)$$

From equations (2.2) and (2.3) we can conclude

$$Q_{S-D/S}(l_1^*, l_2^*) = \min\{Q_{S-D/S}(l_1, l_2) : l_1 \text{ integer}, 2 \leqslant l_1 \leqslant L\}.$$

It only remains to observe that $Q_{S-D/S}(l_1^*, l_2^*)$ is no greater than $Q_{S-D/S}(0, L)$ or $Q_{S-D/S}(1, L-1)$ for $L \geqslant 2$. Thus, l_1^* and l_2^* *are* the optimal choices and the lemma follows.

III. IMPLEMENTATION AND COMPLEXITY OF BINARY SUM CODES

In this section we investigate the practical aspects of implementing on-chip binary sum code error correction. We describe the organization and structure of sum coded RAMs and assess the complexity of implementing different row/column codes in terms of the number of gates needed.

3.1 *Implementing Sum Codes.*

Figure 1 shows the organization of a typical by-1 sum coded dynamic RAM chip . The chip consists of a rectangular array of one bit memory cells, sense amplifiers, refresh circuits, selector and addressing logic, as well as the ECC circuitry. The memory cells are organized as m N-bit word lines, each of which uses $K = k_1 k_2$ cells for data and N-K cells for parity. The total user capacity of the RAM is therefore m*K bits, which implies $\log_2 m\ K$ address bits. The address bits are decoded by the row decoder which transforms $\log_2 m$ row address bits to m word select lines thus selecting one particular N-bit word line. Each read and refresh cycle operates in such a way that all the cells in one complete word line (and hence one complete sum code block) are read out to the sense amps simultaneously. The column decoders transform $L = \log_2 K$ address bits to two groups of select lines, one with k_1 lines and one with k_2 lines. These lines effectively select one column and one row in the sum code. The intersection of these two lines specifies the one data cell in the word line that is being addressed, and the data selector/multiplexer selects and outputs this bit to the ECC unit. The remaining major functional unit in the coded RAM is the V-H code selector block. This unit outputs the vertical and horizontal component code words associated with the bit being addressed. These code words, together with the addressed data bit are fed to the ECC unit which performs any necessary correction and outputs the corrected bit to the user.

The shaded portions in figure 1 show the areas that represent the ECC overhead. There are essentially only three areas of interest. These are the extra memory cells needed to store the parity checks, the selector block needed to 'pull out' the vertical and horizontal component codes of the sum code, and the ECC circuits themselves. It should be particularly noted that no extra column or row decoders are needed, compared with an uncoded memory. The overhead in terms of parity cells has already been determined, and is given by the redundancy of the overall sum code. In the sections that follow, we will estimate the complexity of the two remaining areas.

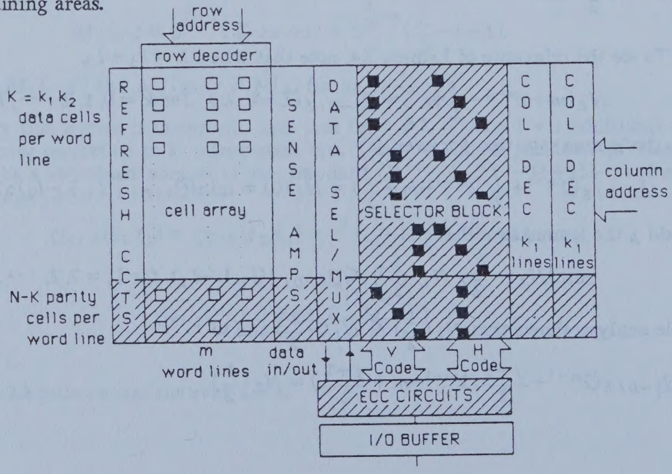


FIGURE 1

3.2 *Selector Block Complexity*

Figure 2 models the circuitry necessary to extract the component horizontal (row), and vertical (column) codewords of the sum code, in terms of multiplexers. It can be seen that n_1 k_2-input muxs are required to extract the horizontal code word in which the addressed data bit lies, and n_2 k_1-input muxs are required to extract the vertical code word. In practice a multiplexer is implemented very simply by a set of 2-input AND gates (one for each MUX input)with their outputs wire-OR'd together. One input of the AND gate is the data input and the other is the controlling input. This input is normally driven via a decoder from the multiplexer select or address lines. However, in our case the select lines are simply the RAM column address lines, and these have already been decoded into 1-out-of-k_1 and 1-out-of-k_2 by the column decoders shown in figure 1. The selector block is therefore simply a collection of 2 input AND gates which together with the column decoder lines acts as the required multiplexers. It therefore follows that a total of $n_1k_2+n_2k_1$ 2-input AND gates are required to extract the horizontal and vertical component codewords. It also follows that the data selector multiplexer requires k_1k_2 = K AND gates; this is obviously constant for fixed K and is also needed in the uncoded case. We can therefore ignore this complexity when comparing different coding schemes, for a given value of K.

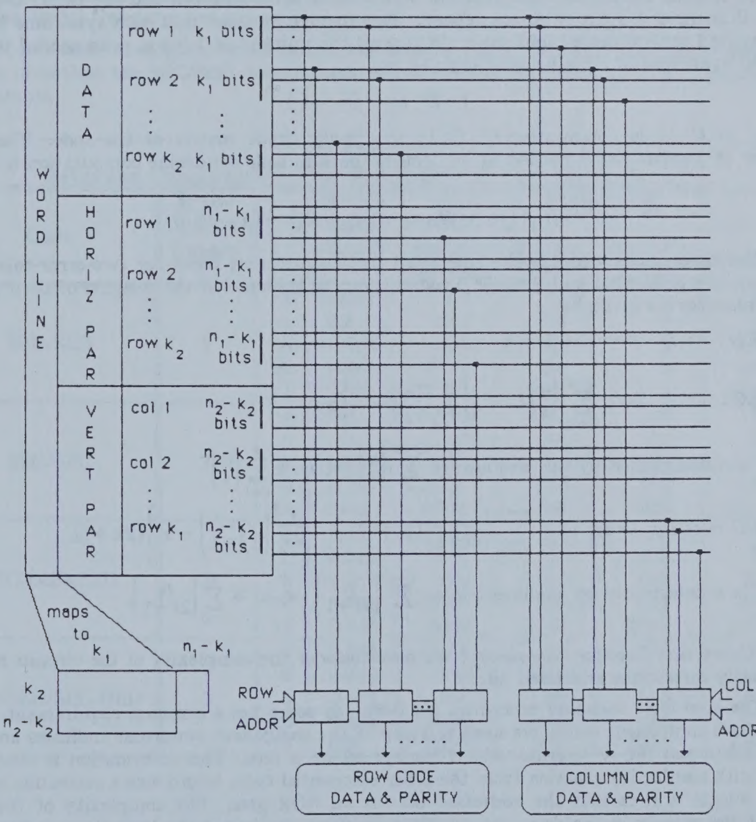

WORD LINE
DATA
row 1 k1 bits
row 2 k1 bits
row k2 k1 bits
HORIZ PAR
row 1 n1 - k1 bits
row 2 n1 - k1 bits
row k2 n1 - k1 bits
VERT PAR
col 1 n2 - k2 bits
col 2 n2 - k2 bits
row k1 n2 - k2 bits
maps to
k1
n1 - k1
k2
n2 - k2
ROW ADDR
COL ADDR
ROW CODE DATA & PARITY
COLUMN CODE DATA & PARITY

Figure 2 Selector Block

3.3 *ECC Complexity*

3.3.1. *Syndrome Complexity:* The first task the ECC hardware must perform is to compute the horizontal and vertical component code syndromes. This is illustrated in figure 3. Each syndrome bit is the XOR sum of various of the selected codeword data bits and the corresponding codeword parity bit. The number of multi-input XOR gates needed is therefore equal to the the number of parity bits in the horizontal code plus the number of checks in the vertical code. In addition, the horizontal XOR gates have potentially $k_1 + 1$ inputs and the vertical XORs have $k_2 + 1$ inputs. However, in practice the multi-input XORs are made up from 2-input XORs, and so an x input XOR requires x-1 2-input XOR gates to implement. The actual number of inputs needed on each XOR depends on the number of data bits involved in the particular syndrome bit computation. This can be found by inspecting the parity check matrix H of the particular code. For example the H matrix of the (7,4) SEC Hamming code is given by:

$$H = \begin{bmatrix} 1\,0\,0 & 0\,1\,1\,1 \\ 0\,1\,0 & 1\,0\,1\,1 \\ 0\,0\,1 & 1\,1\,0\,1 \end{bmatrix}$$

In this case each syndrome bit is the XOR sum of three data bits and one parity bit, indicating that three 4-input gates are needed. This in turn implies that each syndrome bit can be computed with three 2-input gates. In general the number of 2-input gates needed to compute the syndrome of an (n,k) code is given by:

$$||H|| - (n - k)$$

where $||H||$ is the number of 1's in the parity check matrix of the code. The total number of 2-input XORs needed to implement the sum code syndrome computation is therefore

$$||H_1|| + ||H_2|| - (n_1 - k_1) - (n_2 - k_2)$$

The three codes which we're interested in as constituent codes for two-error-tolerating LSC's are the SED, SEC, and SEC-DED codes; it can be shown that the weights of their parity check matrices are given by:

$$SED:\ ||H|| = k + 1 = n$$

$$SEC:\ ||H|| = \sum_{i=1}^{x-1} \binom{n}{i} i + \left[\sum_{j=1}^{x} \binom{n}{j} - n \right] x,$$

where x is determined by the inequalities $\sum_{i=1}^{x-1} \binom{n}{i} < n \leqslant \sum_{i=1}^{x} \binom{n}{i}$.

$$SEC-DED:\ ||H|| = \sum_{i=0}^{x-1} \binom{n}{2i+1} (2i+1) + \left[\sum_{j=0}^{x} \binom{n}{2j+1} - n \right] (2x+1),$$

where x is determined by the inequalities $\sum_{i=0}^{x-1} \binom{n}{2i+1} < n \leqslant \sum_{i=0}^{x} \binom{n}{2i+1}$.

3.3.2. *Correction Decoder Complexity:* We now consider the complexity of the circuits needed to actually correct the addressed bit.

The sum code decoding procedure for different codes has a common requirement. That is, for both component codes, we need to know if the component syndrome indicates an error in the addressed bit, or alternatively if the syndrome is zero. This information is then combined with similar information from the other component code, to produce a correction output signal which will correct the addressed bit via an XOR gate. The complexity of the ECC decoder lies mainly in the detection circuit, and so we can ignore the logic that combines the two component codes. Consider figure 4. This shows a decoder circuit for a three bit syndrome that provides a logic one output if the syndrome indicates an error in a particular bit AND that bit is being addressed. Also, an indication of the all-zero syndrome is output. The look-up table that converts syndromes into error locations is essentially being stored in the interconnections to the AND gates. In the case of the SEC-DED code an indication of a syndrome corresponding to a double error is also needed.

In the case of the SED code, this whole circuit reduces to a piece of wire, as the single bit syndrome provides all the information needed. In general however, k+1 (n-k+1)-input AND gates are needed, plus one more gate for double error detection. Therefore, In terms of 2-input gates, SED requires zero, SEC requires (k+1)(n-k) and SEC-DED requires (k+2)(n-k).

3.4 *Overall Complexity*

Having assessed the complexity of the three main overhead areas needed for sum code ECC in terms of AND gates and XOR gates, we can combine these into a single measure. In CMOS technology the wired-OR AND gates needed can be implemented with 2 transistors, and the XOR gates with 6. Expressing the complexity in terms of transistors then allows us to compare the various coding schemes with a single measure.

IV. Numerical Results

In Table 1 we've listed the possibles rates for for four different classes of LSC's for varying values of L. (In addition to the SEC/SEC and SEC-DED/SED codes examined in section III, we've included data on the SED/SED (i.e., the bidirectional parity check code) and the SEC-DED/SEC-DED codes. It can easily be shown that for these classes of codes, just as for the SEC/SEC codes, the maximum rate is achieved by making the information array square.) In addition, we've listed the complexity measures for the codes which attain these rates.

It can easily seen that, for two-error-tolerance, the SEC-DED/SED code has considerably higher rates than the SEC/SEC codes; in addition, the overall complexities of the two codes are comparable.

Table 1
Rates and Complexities of Several Codes for Varying Values of L

Code	**Error Toler. Capab.**	**L**	$R^*(L)$ **(Best Rate)**	**Complexity**			
				Select. AND's	**Syndr. XOR's**	**Decoder AND's**	**Overall**
SED/SED	1	4	0.667	40	8	0	128
		6	0.8	144	16	0	384
		8	0.889	544	32	0	1280
		10	0.941	2112	64	0	4608
		12	0.97	8320	128	0	17408
SEC/SEC	2	4	0.4	56	18	30	280
		6	0.5	192	36	72	744
		8	0.615	672	76	170	2140
		10	0.727	2432	162	396	6628
		12	0.821	9088	358	910	22144
SEC-DED/SED	2	4	0.471	50	26	50	356
		6	0.615	168	52	108	864
		8	0.744	600	104	238	2300
		10	0.842	2240	224	528	6880
		12	0.908	8608	1056	2580	28712
SEC-DED/SEC-DED	3	4	0.333	64	24	48	368
		6	0.444	208	48	100	904
		8	0.571	704	96	216	2416
		10	0.696	2496	192	476	7096
		12	0.8	9216	416	1056	23040

References

[1] T. C. May and M. H. Woods, "Alpha Particle Induced Soft Errors in Dynamic RAM's", *IEEE Transactions of Electron Devices*, Vol. ED-26, No. 1, pp. 2-9, January 1979

[2] T. Fuja and C. Heegard, "Row/Column Replacement for the Control of Hard Defects in Semiconductor RAM's", Accepted for publication in *IEEE Transactions on Computers*

[3] F. I. Osman, "Error-Correction Technique for Random Access Memories", *IEEE Journal of Solid State Circuits*, Vol. SC-17, No. 5, pp. 877-881, October 1982

[4] T. Mano *et al.*, "Circuit Techniques for a VLSI Memory", *IEEE Journal of Solid State Circuits*, Vol. SC-18, No. 5, pp. 463-469, October 1983

[5] J. Yamada *et al.*, "A Submicron 1 Mbit Dynamic RAM with a 4-Bit-at-a-Time Built-In ECC Circuit", *IEEE Journal of Solid State Circuits*, Vol. SC-19, No 5, pp. 627-633, October 1984

[6] J. E. O'Toole, T. M. Trent, and W. D. Parkinson, "256K Dynamic Error Corrected RAM", memorandum, Micron Technologies, Inc.

[7] T. Fuja, C. Heegard, and R. Goodman, "Some Linear Sum Codes for Random Access Memories", IEEE International Symposium on Information Theory, Brighton, England, June 23-28, 1985

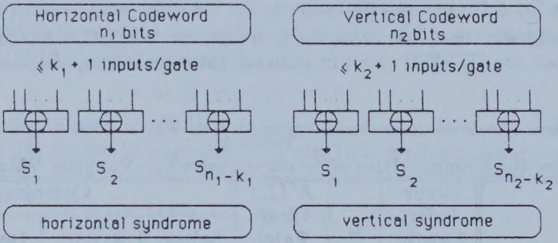

Horizontal Codeword
n_1 bits
$\leqslant k_1 + 1$ inputs/gate
Vertical Codeword
n_2 bits
$\leqslant k_2 + 1$ inputs/gate
S_1
S_2
$S_{n_1-k_1}$
S_1
S_2
$S_{n_2-k_2}$
horizontal syndrome
vertical syndrome

Figure 3. Syndrome complexity

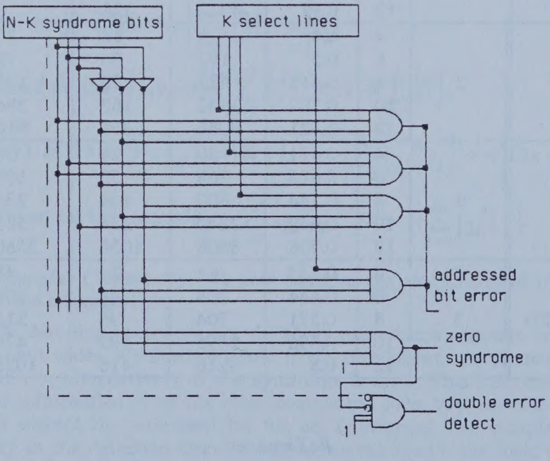

N-K syndrome bits
K select lines
addressed bit error
zero syndrome
double error detect
1
1

Figure 4 ECC complexity

A PRELIMINARY ANALYSIS OF THE HOPFIELD ASSOCIATIVE MEMORY

ROBERT J. McELIECE, EDWARD C. POSNER, EUGENE R. RODEMICH
Jet Propulsion Laboratory and Electrical Engineering Department
California Institute of Technology, Pasadena, California

ABSTRACT

We review Hopfield's recent storage and retrieval algorithms for associative memory, and then demonstrate that if the number of n-bit binary words that are stored exceeds $n/(2\log n)$ (asymptotically), that the memory breaks down.

1. INTRODUCTION

In a very influential recent article, Hopfield [1] introduced a powerful new kind of associative memory, based on his studies of neural networks. Hopfield has demonstrated empirically that this memory is very attractive for many applications, but as yet a good theoretical understanding of its behavior has not been found. We have recently discovered a technique for rigorously analyzing "Hopfield memories," which we introduce in this paper. In the next section (Section 2) we carefully define the Hopfield storage and retrieval algorithms, and then in Section 3, we give our analysis. In Section 3 we use a kind of "signal-to-noise-ratio" analysis borrowed from communication theory, to show that (assuming a result which will be proved in a longer forthcoming paper [2]) that a Hopfield memory, as defined in Section 2, is capable of storing $n/(2\log n)$ binary words of length n, if n is large.

2. THE HOPFIELD ASSOCIATIVE MEMORY

Let $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_M\}$ be a set of M n dimensional binary (± 1) vectors, which are to be stored. We shall call these vectors the *memories*. For each memory $\mathbf{v}_k$ we form the $n \times n$ matrix

$$T_k = \mathbf{v}_k^T \, \mathbf{v}_k - I_n \ ,$$

where I_n denotes the $n \times n$ identity matrix. Thus T_k is just the outer product of $\mathbf{v}_k$ with itself, except that 0's are placed on the diagonal. Now the *Hopfield storage matrix* for the memories $\{\mathbf{v}_1, \ldots, \mathbf{v}_M\}$ is defined as

$$\begin{aligned} T &= \sum_{k=1}^{M} T_k \\ &= \sum_{k=1}^{M} (\mathbf{v}_k^T \, \mathbf{v}_k - I_n) \ . \end{aligned}$$

We assume that once T has been calculated, all other information about the $\mathbf{v}_k$'s will be "forgotten."

Information *retrieval* works as follows. We are given an n dimensional ± 1 vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ (called a *probe*), and wish to find the stored memory $\mathbf{v}_k$ which is "closest" to $\mathbf{x}$ in Hamming distance, using only the storage matrix T. Hopfield's algorithm for doing this is to update the components of $\mathbf{x}$ sequentially using the rule

$$x_i \longleftarrow \text{sign } (\mathbf{x}\, T) \ ,$$

i.e., replace the ith component of $\mathbf{x}$ with the sign (± 1) of the ith component of the vector $\mathbf{x}T$. Hopfield showed that this process is convergent, in the sense that starting with *any vector* $\mathbf{x}$, one will always reach a *fixed vector*, *i.e.*, a vector $\mathbf{x}' = x'_1, x'_2, \ldots, x'_n$ such that

$$x'_i = \text{sign } (\mathbf{x}'T) \ , \quad i = 1, 2, \ldots, n \ .$$

As a simple example, suppose $n = 5$, and that we wish to store the three memories $\mathbf{v}_1 = (+ \ + \ + \ + \ +)$, $\mathbf{v}_2 = (+ \ - \ - \ + \ -)$, and $\mathbf{v}_3 = (- \ + \ - \ - \ -)$. Then we have

$$T_1 = \begin{bmatrix} 0 & + & + & + & + \\ + & 0 & + & + & + \\ + & + & 0 & + & + \\ + & + & + & 0 & + \\ + & + & + & + & 0 \end{bmatrix} \quad T_2 = \begin{bmatrix} 0 & - & - & + & - \\ - & 0 & + & - & + \\ - & + & 0 & - & + \\ + & - & - & 0 & - \\ - & + & + & - & 0 \end{bmatrix} ,$$

$$T_3 = \begin{bmatrix} 0 & - & + & + & + \\ - & 0 & - & - & - \\ + & - & 0 & + & + \\ + & - & + & 0 & + \\ + & - & + & + & 0 \end{bmatrix} ,$$

and so finally

$$T = \begin{bmatrix} 0 & -1 & 1 & 3 & 1 \\ -1 & 0 & 1 & -1 & 1 \\ 1 & 1 & 0 & 1 & 3 \\ 3 & -1 & 1 & 0 & 1 \\ 1 & 1 & 3 & 1 & 0 \end{bmatrix} .$$

Now suppose we are given the probe vector $\mathbf{x} = (+ \ - \ - \ + \ +)$. To update $\mathbf{x}$, we compute $\mathbf{x}T$:

$$\mathbf{x}T = (+4, -3, +4, +4, -2) \ .$$

Thus if we quantize $\mathbf{x}T$ using the rule (1), only the third and fifth components will change. If we select the third component, the new probe will be

$$\mathbf{x}' = (+,-,+,+,+) \ .$$

Now we compute $\mathbf{x}'T$:

$$\mathbf{x}' = (+6,-1,+4,+6,+4)$$

Here we find that the signs of the components of $\mathbf{x}'T$ are the same as those for $\mathbf{x}'$, so the algorithm has converged to a fixed point (unfortunately not one of the memories!). On the other hand, if we had decided to update the *fifth* component of $\mathbf{x}$ first, the new probe vector would have been

$$\mathbf{x}' = (+,-,-,+,-) \ ,$$

and then we would have obtained

$$x'T = (+2,-3,-2,+2,-2) \ ,$$

i.e., no changes in sign, so that we would have converged to the memory $(+,-,-,+,-)$, which is in fact the memory closest to the initial probe $\mathbf{x}$.

This example partially illustrates the possible problems with Hopfield's retrieval algorithm: If we begin with a probe $\mathbf{x}$, the resulting fixed point (a) *may not be one of the memories*; or if it is, (b) *it may not be the nearest memory.* The study of the convergence behavior of Hopfield's algorithm is very complex indeed; in this paper we shall make a start by investigating the much simpler question of *when the memories themselves are fixed points.* This is plainly an important question, since a memory which is not a fixed point can never be "recalled" by the algorithm. We note, for example, that in our example $\mathbf{v}_2$ and $\mathbf{v}_3$ are fixed but $\mathbf{v}_1$ is not, since

$$\mathbf{v}_1 T = (+4,-1,+5,+4,+6)$$

$$\mathbf{v}_2 T = (+2,-3,-2,+2,-2)$$

$$\mathbf{v}_3 T = (-6,+1,-4,-6,-4) \ .$$

3. A "SIGNAL-TO-NOISE RATIO" ANALYSIS (SKETCH)

Without loss of generality, we henceforth assume that the first memory $\mathbf{x}_1$ has all positive components: $\mathbf{x}_1 = (+\ +\ \cdots\ +)$. We model the $n(M-1)$ components of the remaining $M-1$ memories as i.i.d. ±1 (probability 1/2 each) random variables. We are interested in the probability that $\mathbf{x}_1$ is a fixed point of the retrieval algorithm, *i.e.*, that the components of $\mathbf{x}_1 T$ are all positive. To this end, let $\mathbf{x}_1' = \mathbf{x}_1 T$, and note that

$$\begin{aligned} x_{1i}' &= (\mathbf{x}_1 T_1)_i + \sum_{k=2}^{n} (\mathbf{x}_1 T_k)_i \\ &= s_i + z_i \ , \end{aligned}$$

where s_i is the "signal" and z_i is the "noise." Since $\mathbf{x}_1$ is all +'s, we have

$$s_i = n - 1 \ .$$

Our assumptions about the components of $\mathbf{x}_2, \ldots, \mathbf{x}_n$ imply that the noise term z_i is a sum of $M - 1$ i.i.d. random variables, each with mean 0 and variance $n - 1$. Hence if n is fixed and m is large, the normalized noise $z_i \ / \ \sqrt{(n-1)(m-1)}$ approaches a standard normal random variable. It follows then that the probability that the ith component of $\mathbf{x}_1'$ will be negative will be approximately

$$Q\left(\frac{n-1}{\sqrt{(n-1)(m-1)}}\right) \approx Q\left(\sqrt{\frac{n}{m}}\right) \ ,$$

where $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} \, dt$. Thus the *expected number of negative components* in $\mathbf{x}_1'$ is approximately

$$n \, Q\left(\sqrt{\frac{n}{m}}\right) \ .$$

So far our analysis has been fairly rigorous, but now we must omit some fairly difficult calculations and assert without proof that with suitable restrictions, the number of negative components in $\mathbf{x}_1'$ is *approximately Poisson*. (A proof will appear in [2].) Given this, it follows that the probability of *no* negative components, *i.e.*, the probability that $\mathbf{x}_1$ is indeed a fixed point, is given approximately by the expression

$$\beta = \exp\left\{-n \, Q\left(\sqrt{\frac{n}{m}}\right)\right\} \ .$$

Now suppose we require that this probability be a fixed number very near 1, say $\beta = .999999$. Then inverting the preceding expression we get

$$Q\left(\sqrt{\frac{n}{m}}\right) = \frac{\alpha}{n} \ ,$$

where $\alpha = -\log \beta$. This means that

$$m = \frac{n}{Q^{-1}(\frac{\alpha}{n})^2} \ .$$

But for large values of x we have $Q^{-1}(x) \sim \sqrt{2 \log 1/x}$, and so

$$m \sim \frac{n}{2 \log n} .$$

It follows then (modulo our unproved Poisson assumption), that for any value of β (the desired probability of having a given memory fixed) not equal to 0 or 1, the maximum number of memories that can be stored in a Hopfield matrix is asymptotically at most $n/(2 \log n)$. If more than this number are stored, the memories will almost surely not even be fixed points. (And actually we can show that if *less* than this number are stored, not only will they almost surely be fixed points, but also the retrieval algorithm will almost surely converge to the best memory, for any initial probe which is not "too far" from a stored memory.)

References

1. Hopfield, J. J., "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proc. Nat. Acad. Sci. USA, 79 (1982), pp. 2554-2558.
2. McEliece, R. J., E. C. Posner, E. R. Rodemich, S. Venkatesh, "The Capacity of the Hopfield Associative Memory," *in preparation.*

FINDING MINIMAL NESTED POLYGONS

SUBHASH SURI
JOSEPH O'ROURKE

Department of Electrical Engineering and Computer Science
The Johns Hopkins University
Baltimore, MD 21218

ABSTRACT

Let P and Q be two simple polygons with $P \supseteq Q$. A simple polygon K is defined to be nested between P and Q if $P \supseteq K \supseteq Q$. We investigate the problem of determining a nested polygon that has a minimum number of vertices among all polygons nested between P and Q. The minimum vertex nested polygons are characterized in geometrical terms, and an efficient algorithm is developed to find them. The time and space complexity of our algorithm is $O(n^2)$ where n is the total number of vertices of P and Q.

1. Introduction

In this paper we address the problem of finding a minimal polygon nested between two given polygons; here minimality is defined in terms of the combinatorial complexity of the nested polygon. More precisely, given two simple polygons P and Q with $P \supseteq Q$, we seek a polygon K nested between P and Q, $P \supseteq K \supseteq Q$, such that K has a minimum number of vertices among all polygons nested between P and Q. Clearly K is not unique in general; we ask for any representative with the minimum possible number of vertices. Our main result is an $O(n^2)$ algorithm for finding a minimal K, where n is the total number of vertices in P and Q.

Our motivation is from two sources. First, Victor Klee posed the problem of finding a minimal polytope nested between two convex polytopes. Although this problem remains unsolved in dimensions greater than two, we and others [AB] designed an $O(n \log k)$ algorithm for solving Klee's problem in two dimensions, where n is as above and k is the number of vertices of a minimum nested polygon. The generalization to non-convex polygons is a natural generalization of Klee's geometric problem.

Second, there is a practical motivation. If P and Q are similar polygons, one a scaled version of the other, then they can be viewed as the inner and outer envelopes determined by a single polygon with a uniform "tolerance". Most engineering designs include such a tolerance; a tolerance may also derive from position uncertainty in, say, a robotics system. As most polygon algorithm complexities are functions of the number of vertices, it may pay to approximate a complex shape with a simpler one before processing. The general area of polygon approximation has gained considerable attention recently, but usually the approximation criterion has been a continuous measure such as distance, area, or perimeter, rather than the combinatorial complexity of the approximation considered in this paper.

Our computational model, as is usual in this subject, assumes a random-access computer with infinite precision real arithmetic. Our paper is organized in four sections. Section 2 introduces preliminaries and geometric lemmas that lead to an efficient algorithm for a minimal polygon. The algorithm for finding a minimal polygon is given in Section 3. Section 4 concludes with some remarks and directions for further research.

2. Preliminaries

Let P and Q be two given simple polygons such that $P \supseteq Q$. We assume that P and Q do not have a common boundary point. Henceforth, P and Q are used to represent the boundaries of the respective polygons. Let A be the closed region (the annulus) of the plane bounded by P and Q.

In [AB] it was shown that, when P and Q are both convex, a simple greedy strategy leads to a nested polygon that has at most one vertex more than the minimum. In this paper we follow the same general approach. Starting from an arbitrary point in A, a greedy polygon is defined (Lemma 1 through 3). Then we investigate the properties of greedy polygons, showing that they have at most one vertex more than the minimum possible (Lemmas 4 through 8). Finally we show, as our main theorem, that only a linear number of starting points need to be explored in order to find an optimal polygon.

For any point $z \in A$, define its *extreme visible* point on Q, $Q(z)$, as follows. Sweep a line 360° about z and label each line with "P" or "Q" according to which polygon is first intersected by the line. Call this the continuous visibility wheel of z, $w(z)$ (see Fig. 1). Since z is completely surrounded by the given polygons, every angle receives a label. Also, since P and Q are simple, there is at most one contiguous section labeled Q (there could be none). Define $Q(z)$ to be the point of Q corresponding to the most clockwise line of $w(z)$ labeled Q. If several points of Q label this line, $Q(z)$ is set to the point that is most distant from z. $Q(z)$ is undefined if either there is no section labeled Q, or the entire wheel is labeled Q.

$P(z)$ is defined in a similar manner: $P(z)$ is the point of P corresponding to the most counterclockwise line of $w(z)$ labeled P. If several points of P label this line, $P(z)$ is set to the point that is most distant from z. $P(z)$ is undefined exactly when $Q(z)$ is undefined. Clearly, z, $Q(z)$ and $P(z)$ are collinear. Let the longest line segment that is completely contained in A and is directed from z to $Q(z)$ and $P(z)$ be denoted by L_z.

Let $P(z) < Q(z)$ mean that z is closer to $P(z)$ than $Q(z)$ along L_z. $Q(z) < P(z)$ has a similar meaning with the roles of $P(z)$ and $Q(z)$ reversed. Since P and Q do not have a common point, $P(z)$ never equals $Q(z)$. We first establish how L_z is located with respect to P and Q. Due to space limitations we will omit proofs for most of the lemmas; detailed proofs can be found in the full version of this paper [SO].

Lemma 1:

(a) If $Q(z) < P(z)$, then $P(z)$ is the endpoint of L_z, $Q(z)$ is a convex vertex of Q, and Q lies to the left of L_z in the neighborhood of $Q(z)$.

(b) If $P(z) < Q(z)$, then $Q(z)$ is the endpoint of L_z, $P(z)$ is a reflex vertex of P, and P lies to the right of L_z in the neighborhood of $P(z)$.

Proof: Omitted. Q.E.D.

Let l be a maximal line segment (i.e., one that cannot be extended) in A that contains a point $q \in Q$ and a point $p \in P$. Let $q_{\max}$ be the most counterclockwise point, measured along Q's boundary from q, of the set $\{Q(z) \mid z \in l\}$. Let $p_{\max}$ be the most counterclockwise point, measured along P's boundary from p, of the set $\{P(z) \mid z \in l\}$.

The point of extreme visibility for l is defined to be the unique point z^* such that

(1) $Q(z^*) = q_{\max}$, and
(2) $P(z^*) = p_{\max}$.

Points of extreme visibility will be the vertices of the greedy polygon: among the point on l, z^* can "see" farther around Q than any other point. We first must prove that the definition is sound, i.e., that (1) and (2) do not conflict. In the following discussion, the symbols l, z^*, $q_{\max}$ and $p_{\max}$ will be used in accordance with the preceding definition.

Lemma 2: Let $\lambda = \{z \in l \mid Q(z) = q_{\max}\}$. The set λ is closed and connected.
Proof: Omitted. Q.E.D.

We now establish that z^* is well-defined. Let $p_{\max}^{\lambda}$ be the most counterclockwise point, measured along P's boundary from p, of the set $\{P(z) \mid z \in \lambda\}$.

Lemma 3: z^* always exists and is uniquely defined for any l.
Proof: Let a and b be the endpoints of l. Let z_1 and z_2 be the endpoints of λ such that their positions are ordered along l as (a, z_1, z_2, b). Set z^* to be the point of intersection of λ with the line determined by $q_{\max}$ and $p_{\max}^{\lambda}$. Clearly, z^* is a unique point. Property (1) of the definition

of z^* trivially follows from the fact that $z^* \in \lambda$. In order to establish the property (2), we must show that $p_{\max} = p^{\lambda}_{\max}$. The proof proceeds by analyzing the four cases arising from the relative positions of $P(z_i)$ and $Q(z_i)$ along L_{z_i}, $i \in \{1,2\}$. We will detail the proof of only two cases with $Q(z_1) < P(z_1)$; the remaining two cases with $P(z_1) < Q(z_1)$ are handled similarly.

Case 1: $Q(z_1) < P(z_1)$ and $P(z_2) < Q(z_2)$.

$Q(z_1) < P(z_1)$ implies that $Q(z_1)$ is a convex vertex. Since $P(z_2) < Q(z_2)$ and $Q(z_2) = Q(z_1) = q_{\max}$ is convex, $z_2Q(z_2)$ can be extended further in A. If $z_2Q(z_2)$ extended intersects Q first, then the hypothesis that $Q(z_2) = q_{\max}$ is contradicted. If it intersects P first, then the hypothesis that $P(z_2) < Q(z_2)$ is contradicted. Hence, Case 1 cannot occur.

Case 2: $Q(z_1) < P(z_1)$ and $Q(z_2) < P(z_2)$.

(Fig. 2) Since $Q(z_1) < P(z_1)$, $Q(z_1) = q_{\max}$ is a convex vertex. It is clear that the most clockwise point, measured along P from p, of the set $\{P(z) \mid z \in [a, z_1)\}$, is at most $P(z_1)$. Therefore, if $\{z \mid z \in (z_2, b]\} = \emptyset$ the claim that $p^{\lambda}_{\max} = p_{\max}$ follows immediately. Otherwise, if L_{z_2} contains a reflex vertex $p_r \in P$ between z_2 and $q_{\max}$, then the most counterclockwise point of the set $\{P(z) \mid z \in (z_2, b]\}$ is at most $P(z_2)$, and the claim that $p^{\lambda}_{\max} = p_{\max}$ follows again. In order to show the existence of p_r, consider a point $z_3 \in (z_2, b]$ in the neighborhood of z_2. Clearly, $z_3 \in l - \lambda$. P cannot cross $z_3q_{\max}$ because that would imply the existence of $p_r \in P$ on L_{z_2}. Since Q is simple and does not cross l, L_{z_2} and P, the segment $z_3q_{\max}$ cannot be crossed by Q. The segment $z_3q_{\max}$ lies completely in A and, therefore, $Q(z_3) = q_{\max}$. Hence by the definition of λ, $z_3 \in \lambda$, which is a contradiction to the hypothesis that $z_3 \in l - \lambda$. The claim is, therefore, established. Hence, $p_{\max} = p^{\lambda}_{\max}$.

This completes our proof that $p_{\max} = p^{\lambda}_{\max}$. The existence and uniqueness of z^* is, therefore, established. Q.E.D.

We now define the greedy polygon in terms of the z^* points. Two line segments l_1 and l_2 are considered *overlapping* if and only if their common intersection is a non-degenerate segment. Define a greedy polygon $G_a = (v_1, v_2, \ldots, v_k)$, for a point $a \in A$, as follows. The first edge of G_a overlaps L_a. The i^{th} edge, $1 \le i \le k$, overlaps L_{v_i} where $v_i = z^*_{i-1}$ is the i^{th} vertex of G_a, and z^*_{i-1} is the extreme visibility point for $L_{v_{i-1}}$. The vertex v_1 is determined by the intersection of L_{v_k} and L_a where k is the number of vertices of G_a. The vertex v_1 does not necessarily coincide with z^*_k (Fig. 3). Note that a itself may not belong to G_a.

The next two lemmas characterize the local structure of greedy polygons.

Lemma 4: Let $G_a = (v_1, v_2, \ldots, v_k)$ be a greedy polygon. $P(v_i) < Q(v_i)$ if and only if v_{i+1} is a reflex vertex.

Proof: Omitted. Q.E.D.

Lemma 5: Let $G_a = (v_1, v_2, \ldots, v_k)$ be a greedy polygon. Each vertex v_i, $2 \le i \le k$, must satisfy the following:

(i) If $Q(v_i) < P(v_i)$ then either L_{v_i} contains a reflex vertex $p_r \in P$ and a convex vertex $q_c \in Q$ such that $p_r < q_c$, or v_i lies on P.

(ii) If $P(v_i) < Q(v_i)$ then either L_{v_i} contains a reflex vertex $p_r \in P$ and a convex vertex $q_c \in Q$ such that $q_c < p_r$, or v_i lies on Q.

Proof: Omitted. Q.E.D.

Next we show that a greedy polygon has at most one vertex more than the minimum. For this purpose, we associate disjoint subsets of A, one per edge of a greedy polygon.

Let $G_a = (v_1, v_2, \ldots, v_k)$ be a greedy polygon. Let v_iv_{i+1} be an edge of G_a with $P(v_i) < Q(v_i)$ and $v_i = z^*_{i-1}$. Note that the second restriction may only rule out v_1v_2 from the consideration. Let $q \in Q$ be the point closest to v_i that lies on L_{v_i}. Define R_i to be the closed region that lies to the left in the traversal from q to $Q(v_i)$ along L_{v_i} followed by a clockwise traversal along Q from $Q(v_i)$ to q and excludes the point q.

Similarly, if $Q(v_i) < P(v_i)$, let $p \in P$ be the point closest to v_i that lies on L_{v_i}. Define R_i to be the region that lies to the left in the traversal from p to $P(v_i)$ along L_{v_i} followed by a clockwise traversal along P from $P(v_i)$ to p and excludes p. These regions are illustrated in Figure 3. Note that the existence of q in the former case, and that of p in the latter follows immediately from Lemma 5. R_i is associated with the edge $v_i v_{i+1}$, which partially overlaps its boundary. The following two lemmas concern themselves with these regions.

Lemma 6: Let $G_a = (v_1, v_2, \cdots v_k)$ be a given greedy polygon. Let R_i and R_j be the regions associated with the edges $v_i v_{i+1}$ and $v_j v_{j+1}$, respectively. The regions R_i and R_j are disjoint, for $i \neq j$, $2 \leq i, j \leq k-1$.
Proof: Omitted. Q.E.D.

Lemma 7: Let R_i be a region in A. Any polygon nested between P and Q must intersect R_i.
Proof: Omitted. Q.E.D.

The following lemma establishes the sub-optimality of a greedy polygon.

Lemma 8: A greedy polygon $G_a = (v_1, v_2, \ldots, v_k)$ has at most one vertex more than the minimum.
Proof: Let R_i be the region associated with the edge $v_i v_{i+1}$, $2 \leq i \leq k$. Note that $v_1 v_2$ may not have a region associated with it if v_1 is distinct from z_k^*. By Lemma 6, these regions are disjoint, and by Lemma 7 any polygon nested between P and Q intersects each of these $k-1$ regions. Since two points lying outside a region R_i, $2 \leq i \leq k$, cannot be joined through R_i without introducing a vertex in it, a polygon nested between P and Q must have at least one vertex in each region R_i, $2 \leq i \leq k$. A minimal polygon, therefore, will have at least $k-1$ vertices, and the lemma is established. Q.E.D.

The next step is to show that it is sufficient to consider the greedy polygon only.

Lemma 9: Let $K = (v_1, v_2, \cdots, v_k)$ be a minimal polygon. The greedy polygon $G_{v_1} = (w_1, w_2, \ldots, w_k)$ has minimal number of vertices.
Proof: We show that K and G_{v_1} "interleave" and, therefore, have same number of vertices. The first edge of G_{v_1}, $w_1 w_2$ overlaps L_{v_1}. Since $Q(v_1)$ and $P(v_1)$ lie on L_{v_1}, K must cross L_{v_1} at other than v_1. In fact, L_{v_1} must intersect $v_2 v_3$, because otherwise a nested polygon with fewer vertices than K can be constructed by "shortcutting" along L_{v_1}. Since $w_i = z_{i-1}^*$, L_{w_i} must intersect with $v_{i+1} v_{i+2}$, $1 \leq i \leq k-1$, where the indices are modulo k, by the same argument. Finally, L_{w_k} intersects with $v_1 v_2$ at a point which is w_1 by the definition of the greedy polygon G_{v_1}. The preceding argument has established that K and G_{v_1} "interleave", and the lemma is proved. Q.E.D.

Before proving the main characterization theorem of this paper, we need some results on the special case when the nested polygon is convex; the two given polygons, P and Q, may be non-convex. This special case, though different from that solved in [AB], has many similar properties that we will review in brief. Our discussion will mainly focus on the details that are relevant to the development of this paper; [AB] contains a more detailed exposition.

For any point $z \in P$, the greedy polygon $G_z = (v_1, v_2, \cdots, v_k)$ is called a *supporting polygon*, S_z, if

(a) G_z is convex, and
(b) all vertices of G_z except perhaps v_1 lie on P.

Let $S_z = (v_1, v_2, \cdots, v_k)$ be a supporting polygon. Since all vertices of S_z are convex, Lemma 1 and 4 imply that $Q(v_i)$, $1 \leq i \leq k$, is a convex vertex of Q. The Q-contact for the edge $v_i v_{i+1}$, $1 \leq i \leq k$, is defined to be $Q(v_i)$. The P-contact for the edge $v_i v_{i+1}$, $2 \leq i \leq k-1$, is defined to be the edge of P that contains $v_i = P(v_{i-1})$. Note that the P-contact is not defined for the edge $v_1 v_2$ if v_1 does not lie on P. If v_i coincides with a vertex, say $p_j \in P$, then the P-

contact is chosen to be the more counterclockwise edge, that is, $p_j p_{j+1}$. The contacts for all the edges of S_z together define the *set of contacts* for S_z.

Let S_{z_o} be an initial supporting polygon. Consider sliding a point z counterclockwise along P from z_o. The set of contacts for S_z will vary as z slides along P. A *contact change* is reported at a position $z=z_c$ if S_{z_c} has a different set of contacts than S_z for a position of z in the clockwise neighborhood of z_c. The point z_c is called a contact change point. We shall assume throughout that not more than one contact changes at a contact change point. It is clear that if z_c is a contact change point, the following must hold for S_{z_c} (Fig. 4):

(a) If a Q-contact changes at z_c, that is, the Q-contact for an edge $v_i v_{i+1} \in S_z$ changes from q_1 to q_2, then $v_i v_{i+1}$ for S_{z_c} overlaps L_{q_1}.

(b) If a P-contact changes at z_c, that is, the P-contact for an edge $v_i v_{i+1} \in S_z$ changes from p_1 to p_2, then $v_i v_{i+1}$ for S_{z_c} overlaps L_{p_1}.

The following lemma is taken from [AB], and is stated here without proof.

Lemma 10: Let $S_z=(v_1, v_2, \cdots, v_k)$ be a supporting polygon nested between P and Q. Let x_0 be a parametric representation of the position of z, expressed as a distance along z's contact edge. Similarly, let x_i be the representation of v_i, $2 \leq i \leq k$. The functional relationship, f, between x_i and x_0 is

$$x_i = f_i(x_0) = (a_1 + a_2 x_0)/(a_3 + a_4 x_0)$$

where a_1, a_2, a_3 and a_4 are constants that depend only on the contacts between z and v_i.

Proof: Omitted. (See [AB]).Q.E.D.

Since z, v_1 and v_2 all lie on L_z, the position of v_1 along L_z is only of secondary importance for computing the greedy polygon G_z. Hence, the function f_i relates the position of the vertex v_i, $2 \leq i \leq k$, to that of z rather than to v_1.

Although Lemma 9 implies that a minimal polygon can be found by examining all greedy polygons, there are an infinite number of possibilities for the choice of the first vertex, $a \in A$, from which to construct a greedy polygon G_a. A brute force search though these possibilities is, therefore, out of question. The characterization that permits this search to be discretized is given in the following theorem, which establishes that only vertices of P and Q need to be examined as starting points. Since there are only n such vertices, an efficient algorithm can be constructed to examine them all.

Theorem 1: There exists a greedy polygon G_z with minimal number of vertices, where z is vertex of P or Q.

Proof: Lemma 9 guarantees that there is a minimal greedy polygon $K=(v_1, v_2, ..., v_k)$. We will assume that K does not satisfy the theorem, and then modify it without increasing its number of vertices to one that does satisfy the claim. We will detail the proof for only non-convex K.

Case 1: K is non-convex (Fig. 5).

Let v_{i+1} be a reflex vertex immediately preceded by a convex vertex v_i. First suppose that $v_i = z^*_{i-1}$; that is, either $i \neq 1$ or v_1 coincides with z^*_k. By Lemma 4, $P(v_i) < Q(v_i)$ and by Lemma 1, $P(v_i)$ is a reflex vertex. Let $P(v_i)=p_r$. If a convex vertex $q_c \in Q$ lies on L_{v_i} between v_i and p_r, G_{q_c} clearly meets the claim of the theorem. Therefore, assume to the contrary that no such vertex lies between v_i and p_r. Let z be a point in the clockwise neighborhood of v_i along $v_{i-1}v_i$. The line segment zp_r is not crossed by either P and Q. Since $Q(z)$ lies farther than $Q(v_i)$, measured along Q from $Q(v_{i-1})$, it contradicts the assumption that $v_i = z^*_{i-1}$. Hence, the theorem is established for the case when $v_i = z^*_{i-1}$.

Now suppose that K has only one reflex vertex, v_2, and the preceding convex vertex $v_1 \neq z^*_k$. Again, it follows from Lemma 4 that $P(v_1) < Q(v_1)$. Consider $L_{z^*_k}$. If $L_{z^*_k}$ intersects any edge $v_i v_{i+1}$ for $i > 2$, let v'_i be the point of intersection. The nested polygon

$K' = (z_k^*, v_i', v_{i+1}, \cdots, v_k)$

has fewer vertices than K; a contradiction to the assumed minimality of K. Hence, $L_{z_k^*}$ intersects v_2v_3 only, and the point of intersection lies in the open segment (v_2, v_3). Let v_2' be the point where $L_{z_k^*}$ intersects with v_2v_3. Consider the nested polygon $K' = (z_k^*, v_2', v_3, \ldots, v_k)$. It has the minimum number of vertices, k. By Lemma 9, $G_{z_k^*} = (w_1, w_2, \cdots, w_k)$ also has minimum number of vertices. Since $P(z_k^*) < Q(z_k^*)$, $w_2 \in G_{z_k^*}$ is a reflex vertex that is immediately preceded by a convex vertex, w_1. $G_{z_k^*}$, therefore, falls into the case discussed above, and the theorem follows.

Case 2: K is convex.

The proof for this case is omitted. Q.E.D.

To sum up, this section has established that a minimal polygon can be found by examining all the greedy polygons G_z, where z is a vertex of either P or Q. Our next task, therefore, is to design an efficient algorithm for constructing a greedy polygon. Section 3 shows that after an $O(n^2)$ preprocessing, G_z, where z is a vertex of P or Q, can be computed in $O(n)$ time.

3. Algorithm for Minimal Polygons

Our algorithm for finding a minimal polygon that is nested between P and Q constructs the greedy polygons for each vertex of P and Q. By Theorem 1, a minimal polygon is guaranteed to be found among these polygons. Given $z \in A$, the corresponding greedy polygon, G_z, is constructed by fixing the first edge to be L_z, and then successively computing $v_{i+1} = z_i^*$ and $L_{v_{i+1}}$, $1 \leq i \leq k-1$. Although v_1 is not determined until the last edge, L_{v_k}, is computed, all the other vertices, namely $v_2 = z_1^*$, $v_3 = z_2^*$, ..., $v_k = z_{k-1}^*$ can be determined in that order. The following algorithm, Algorithm-A, gives a pseudo code for finding a minimal polygon:

Algorithm-A: Finding a minimal polygon nested between P **and** Q

```
Preprocessing [Algorithm-B];
for $z$ = each vertex of $P$ and $Q$ do begin
    {Compute $G_z$}
    Let the first edge be $L_1 = L_z$;
    $i \leftarrow 1$;
    repeat
        $i \leftarrow i+1$;
        $v_i \leftarrow z_{i-1}^*$ [Algorithm-C];
    until $L_{v_i}$ intersects $L_1$;
    $v_1 \leftarrow$ point of intersection between $L_i$ and $L_1$;
end {for}
```

Our next task is to compute z_i^*'s efficiently. Since z_i^* can lie anywhere along L_{v_i}, an exhaustive search is impossible. Instead, we use the vertex to vertex visibility graph V_G among the vertices of P and Q to perform a discrete search along L_{v_i}. V_G has vertices of P and Q as nodes, and two nodes, v_i and v_j, have a visibility edge between them if and only if the line segment v_iv_j lies completely in A (recall A is closed). V_G can be computed in $O(n^2)$ (see [We]). Once V_G is available, we use it to compute the following information:

(a) $P(z)$, where z is a vertex of P or Q,

(b) $Q(z)$, where z is a vertex of P or Q,

(c) $succ(z) = p$, where z is a vertex of P, and p is the most clockwise *vertex* of P that labels a line with P in the visibility wheel of z, $w(z)$,

(d) *pred*$(z)=q$, where z is a vertex of Q, and q is the vertex of Q whose extreme visible point on Q is z, that is, $Q(q)=z$.

The following algorithm, Algorithm-B, describes this computation in pseudo code. The set of neighbors of v_i in V_G defines the *discrete visibility wheel* of v_i, namely $W(v_i)$.

Algorithm-B: Preprocessing

```
Construct $V_G$;
for $z$=each vertex of $P$ or $Q$ do begin
    Compute $W(z)$ from $V_G$;
    Let $q_i$ be the most clockwise vertex of $Q$ in $W(z)$;
    Let $p_j$ be the most counterclockwise vertex of $P$ in $W(z)$;
    if $z \in P$ then $succ(z) \leftarrow p_j$;
    if $zq_i$-extended first intersects $Q$ then
        $Q(z) \leftarrow$ point where $zp_j$-extended intersects $Q$
    else begin
        $Q(z) \leftarrow q_i$;
        if $z \in Q$ then $pred(q_i) \leftarrow z$;
    end
end {for}
end {Algorithm-B}
```

Algorithm-B has $O(n^2)$ time complexity, since the **for** loop is executed only n times, and each execution of the loop takes only $O(n)$ time. $P(z)$, $Q(z)$, $succ(z)$ and $pred(z)$ for each vertex of P and Q can, therefore, be computed in $O(n^2)$ time.

The following algorithm, Algorithm-C, describes how to compute z^* for the line segment L_z. The algorithm, in fact, computes L_{z^*}. We give an algorithm only for the case where $Q(z) < P(z)$. The case where $P(z) < Q(z)$ is similar. The two pointers, p and q, are used to traverse the boundaries of P and Q respectively. For a vertex $q \in Q$, let z_L be the point on L_z closest to $Q(z)$ that is visible from q. Similarly, let z_R be the point on L_z closest to $P(z)$ that is visible from q. Initially, q is set to $Q(z)$, z_L to $Q(z)$ and z_R to $P(z)$. Therefore, $z_L < z_R$ at the start. As q advances on Q, z_L and z_R move on L_z in opposite directions. z^* is located at the crossover point.

Algorithm-C: Computing z^*

```
Initialization Step:
  $q \leftarrow Q(z)$;
  $p \leftarrow p_j$ where $p_{j-1}p_j$ is the edge of $P$ containing $P(z)$;
  $z_L \leftarrow Q(z)$;      $z_R \leftarrow P(z)$;
  finished $\leftarrow$ false,
General Step:
while not finished do begin
    if $L_q$ does not intersect $L_z$ between $z_L$ and $z_R$ then
        $L_{z^*} \leftarrow$ longest segment in $A$ directed from $z_R$ to $q$;
        finished $\leftarrow$ true;
    else if $P(q) < Q(q)$ then
        $L_{z^*} \leftarrow L_q$;
        finished $\leftarrow$ true;
    else {$Q(q) < P(q)$}
        $z_L \leftarrow$ point of intersection of $L_q$ with $L_z$;
        {advance the $q$ pointer}
        $q \leftarrow Q(q)$;
        {advance the $p$ pointer}
        while $Q(q)$ is more counterclockwise than $Q(succ(p))$
```

```
11.                 do $p \leftarrow succ(p)$;
                end {while}
12.             if $qp$ extended in $A$ intersects $L_z$ at $x \in (z_L, z_R)$
13.                 then $z_R \leftarrow x$;
    end {while not finished}
```

A sample execution of Algorithm-C is illustrated in Fig. 6. The proof of correctness of Algorithm-C is omitted.

Analysis of the time complexity of Algorithm-C

Step 3 only requires traversing along P to determine $P(z_R)$, starting from $P(z)$, and can be executed in time proportional to the the number of vertices of P between $P(z)$ and $P(z_R)$. Step 6 requires only constant amount of time. The pointer q is advanced only from $Q(z)$ to $Q(z^*)$, and similarly for the pointer p. It is clear, therefore, that the time spent by algorithm Algorithm-C in computing z^* is proportional to the vertices of Q that lie between $Q(z)$ and $Q(z^*)$, and the vertices of P that lie between $P(z)$ and $P(z^*)$.

It follows from the complexity analysis of Algorithm-C that a greedy polygon G_z, $z \in A$, can be computed in $O(n)$ time by successively computing $L_1, L_2, \ldots, L_k$. Consequently, a minimal polygon can be found by constructing a greedy polygon for each vertex of P and Q by Algorithm-A in $O(n^2)$ time. Finally, since the visibility graph computed in Algorithm-B takes $O(n^2)$ space, our space requirement is also $O(n^2)$.

4. Discussion and Extensions

We have presented an $O(n^2)$ time and space algorithm for finding a simple minimum vertex nested polygon. A greedy polygon that has at most one vertex more than the minimum can be constructed in $O(n)$ time after $O(n^2)$ preprocessing. We have also developed an algorithm (not presented here) for finding a single greedy polygon in $O(n \log n)$ time and $O(n)$ space, with no preprocessing. By Lemma 8, this gives a fast sub-optimal algorithm.

Considerable simplification results if the polygons are restricted to be orthogonal: then we can show that a minimal nested polygon can be found in $O(n \log n)$ time in the general case, and $O(n)$ time if P and Q are orthogonally convex.

Acknowledgements

This work was partially supported by NSF grant DCR-83-51468.

5. References

[AB] A. Aggarwal, H. Booth, J. O'Rourke, S. Suri, C. K. Yap. "Finding Convex Minimal Nested Polygon", *Symposium on Computational Geometry,* pp. 296-304 (June 1985).

[CY] J.S. Chang and C.K. Yap. "A Polynomial Solution for Potato-Peeling and Other Polygon Inclusion and Enclosure Problems," *Foundations of Computer Science,* pp. 408-417 (1984).

[DA] N.A.A. De Pano and A. Aggarwal. "Finding Restricted k-envelopes For Convex Polygons," *Proc. 22nd Allerton Conference,* pp. 81-90 (1984).

[DB] B. Dori and M. Ben-Bassat, "Circumscribing a Convex Polygon by a Polygon of fewer sides with Minimum Area Addition," *CVGIP,* 24 pp. 131-159 (1983).

[KL] V. Klee and M.C. Laskowski. "Finding the Smallest Triangle Containing a Given Convex Polygon," *J. of Algorithms,* to appear (1985).

[OR] J. O'Rourke. "Finding Minimal Enclosing Boxes," JHU, Tech. Report (1984).

[SO] S. Suri and J. O'Rourke. "Finding Minimal Nested Polygons," Technical Report, The Johns Hopkins University (May, 1985).

[To] G. Toussaint. "Solving Geometric Problems with Rotating Calipers," *Proc. IEEE MELECON '83,* (May 1983).

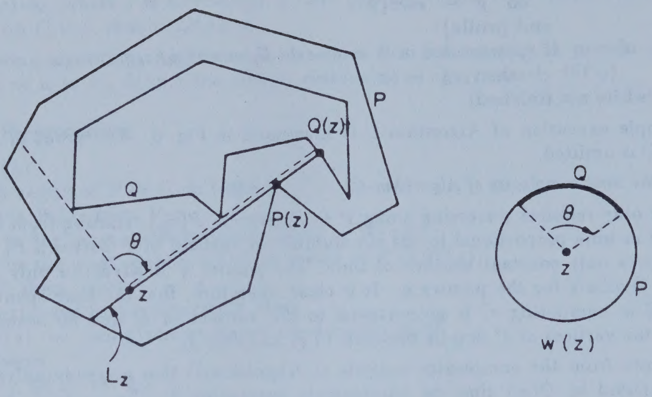

P
Q(z)
Q
θ
P(z)
z
L_z
Q
θ
z
P
w(z)

Figure 1.

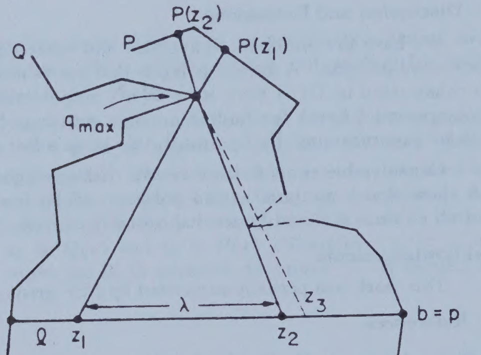

P(z_2)
Q
P
P(z_1)
q_max
λ
z_3
a = q
Q
z_1
z_2
b = p

Figure 2.

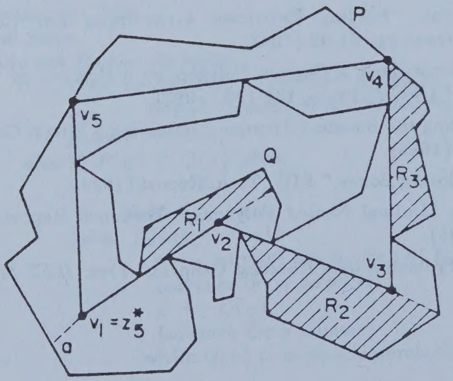
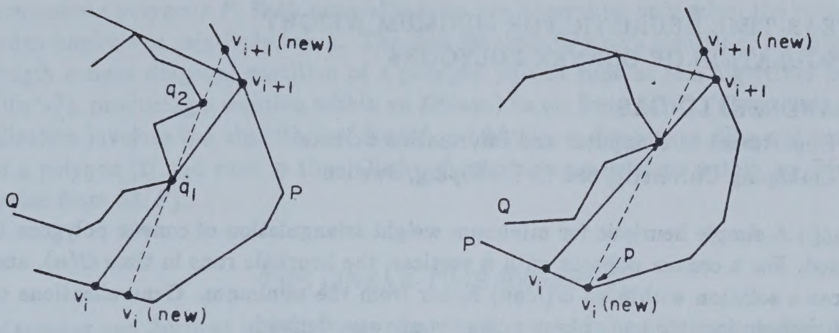

P
v_4
v_5
Q
R_3
R_1
v_2
v_3
R_2
v_1 = z*_5
a

Figure 3.


v_{i+1} (new)
v_{i+1}
q_2
q_1
Q
P
v_i
v_i (new)
v_{i+1} (new)
v_{i+1}
Q
P
p
v_i
v_i (new)

(a) (b)

Figure 4.

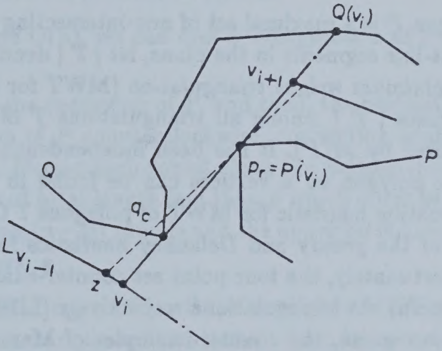
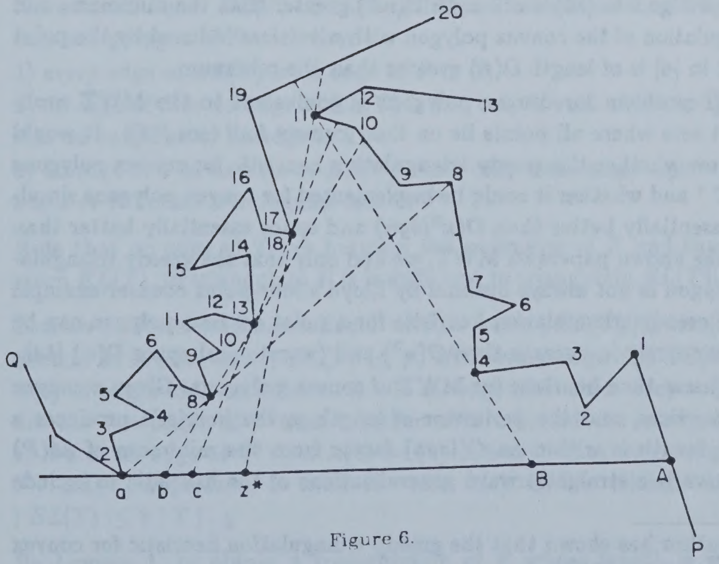

Q(v_i)
v_{i+1}
P
Q
p_r = P(v_i)
q_c
L_{v_{i-1}}
z
v_i

Figure 5.


20
19
11
12
13
10
9
8
16
17
18
14
15
7
6
12
13
11
10
5
9
6
4
3
1
5
7
8
4
2
3
Q
1
2
a
b
c
z*
B
A
P

q	p	z_L	z_R
q_2	p_1	a	A
q_8	p_1	a	A
q_{13}	p_1	b	A
q_{18}	p_4	c	B

Figure 6.

A LINEAR-TIME HEURISTIC FOR MINIMUM WEIGHT TRIANGULATION OF CONVEX POLYGONS

ANDRZEJ LINGAS
Department of Computer and Information Science
Linköping University, 581 83 Linköping, Sweden

Abstract : A simple heuristic for minimum weight triangulation of convex polygons is presented. For a convex polygon with n vertices, the heuristic runs in time $O(n)$, and produces a solution within an $O(log n)$ factor from the minimum. Generalizations of the heuristic to include non-convex polygons are also derived.

INTRODUCTION

A *triangulation* of a polygon P is a maximal set of non-intersecting internal diagonals of P. For a set T of straight-line segments in the plane, let $\mid T \mid$ denote the total length of the segments in T. A *minimum weight triangulation* (MWT for short) of P is any triangulation which minimizes $\mid T \mid$ among all triangulations T of P. The length of a MWT of P will be denoted by $M(P)$. It has been independently shown in [3] and [5] that MWT of a simple polygon on n vertices can be found in time $\Omega(n^3)$. Does there exist faster, approximation heuristic for MWT of polygons ? One could consider the polygon counterparts of the greedy and Delauney heuristics for MWT of planar point sets [10,11, 12]. Unfortunately, the four-point set counter-examples of Lloyd for the greedy and Delauney point set triangulations respectively [Ll78] work also in the (convex) polygon case. Even worse, the counter-examples of Manacher and Zorbrist [13] for the greedy point set triangulation, and of Kirkpatrick [4] for the Delauney point set triangulation also hold in the polygon case. By proofs in [13] and [4], we can easily conclude that the greedy triangulation of a non-convex polygon with n vertices induced by the point set from Fig. 2 in [13] is of length $\Omega(n^{\frac{1}{2}})$ greater than the minimum, and the Delauney triangulation of the convex polygon with n vertices induced by the point set from Theorem 1 in [4] is of length $\Omega(n)$ greater than the minimum.

Note that the MWT problem for convex polygons is equivalent to the MWT problem for planar point sets where all points lie on their *convex hull* (see [12]). It would be interesting to know whether the greedy triangulation heuristic for convex polygons approximates MWT * and whether it could be implemented for convex polygons simultaneously in time essentially better than $O(n^2 log n)$ and space essentially better than $O(n^2)$ (see [3]). In the known papers on MWT, we find only that the greedy triangulation of a convex polygon is not always optimal by Lloyd's four-point counter-example [Ll78], and that the greedy triangulation heuristic for a point set or a polygon can be implemented simultaneously in average time $O(n^2)$ and (worst-case) space $O(n)$ [14].

Here, we present a linear-time heuristic for MWT of convex polygons. Given a convex polygon P with n vertices, and the perimeter of length p, the heuristic produces a triangulation whose length is within an $O(log n)$ factor from the minimum of $M(P)$ and p. Next, we derive two straightforward generalizations of the heuristic to include non-convex polygons P. Both generalizations are interesting only when the number of reflex angles r is relatively small. The first involves Greene's algorithm for minimum length convex diagonal partition of a polygon [6] and runs as this algorithm in time $O(n^2r^2)$, producing a solution within an $O(logn)$ factor from $M(P)$. The second generalization involves the algorithm of Hertel and Mehlhorn for convex diagonal partition of a polygon [2] and runs in time $O(nlogn)$, producing a solution within an $O(rlogn)$ factor from $M(P)$.

---

* Recently, the author has shown that the greedy triangulation heuristic for convex polygons approximates MWT [9].

THE LINEAR-TIME HEURISTIC

Manacher and Zorbrist in [13] observed a simple way of triangulating a regular 2^k-gon with diameter d by using diagonals of total length $O(dk)$. Lingas in his dissertation generalized the above observation as follows:

Fact 1 : [8]: In time $O(n)$, we can find a triangulation of P of length $\leq p\lceil log_2(n-2)\rceil$.

Sketch: We follow the perimeter of P, and then, the perimeter of the resulting, current convex sub-polygon of P counterclockwise, connecting consecutive, non-adjacent vertices by diagonals. During each round along the perimeter of the current sub-polygon, we draw diagonals of total length less than p which cover at least $\lfloor \frac{n}{2} \rfloor$ vertices. Hence, we have at most $\lceil log_2(n-2)\rceil$ rounds and the upper bound on the length of the produced triangulation follows. ∎

Fact 1 yields a satisfactory solution if the length of a MWT of P is $\Omega(p)$. One can easily construct a counter-example to the heuristic from the proof of Fact 1, in the form of a very flat convex polygon with two long spikes on the left and right side such that the ratio between the length of the triangulation produced by the heuristic and that of a MWT can be arbitrarily large. To get rid of the dependency on the p factor, we define the *external layer* $EL(T)$ of a triangulation T of a non-necessarily convex polygon P, as a sub-polygon of P such that:

1) every edge of $EL(T)$ is an edge of P or T,

2) an edge of P is an edge of $EL(T)$ if and only if each of the endpoints of the edge is also an endpoint of an edge in T,

3) an edge of T is an edge of $EL(T)$ if and only if the edge together with two adjacent edges of P bound a triangular face in $P \cup T$.

Note that no edge of T lies between the perimeter of P and that of $EL(T)$. Hence, given $EL(T)$, to triangulate P it is sufficient to triangulate $EL(T)$. Moreover, we have:

Lemma 1: For any triangulation T of P, it holds $\mid EL(T) \mid \leq min(2\mid T\mid, p)$.

Sketch: It is clear that $\mid EL(T)\mid \leq p$. With every edge e of $EL(T)$, associate the set $A(e)$ of edges of T as follows: if e is an edge of P then $A(e) = \{e_1, e_2\}$ where e_1, e_2 are the two edges of T that together with e bound the triangular face adjacent to e; if e is an edge of T then $A(e) = \{e\}$. Observe that every edge in $T - EL(T)$ or $T \cap EL(T)$ appears in at most two sets $A(e)$. By the triangle inequality, we obtain $\mid EL(T)\mid \leq 2\mid T\mid$. ∎

By Lemma 1, to obtain a triangulation of P whose length is within a logarithmic factor from $M(P)$ it is sufficient to find $EL(T)$ where T is a MWT of P and then to triangulate $EL(T)$ using the procedure from Fact 1. However, finding $EL(T)$ does not seem easier than finding a MWT of P. Therefore, instead of constructing the external layer of a MWT of P, we shall construct a shortest, so called, *candidate for an external triangulation layer* in P. The set of all such candidates will be denoted by $CEL(P)$. A set C of edges and diagonals of P is in $CEL(P)$ if and only if C is a simple polygon whose vertices in clockwise order form a subsequence of vertices of P in clockwise order such that for every vertex v of P that is not a vertex of C, the two vertices of P adjacent to v are also vertices of C.

Clearly, given a polygon C in $CEL(P)$, to triangulate P it is sufficient to triangulate C. Moreover, for any triangulation T of P, $EL(T)$ is in $CEL(P)$.

By the two observations and Lemma 1, it is sufficient to apply the heuristic from the proof of Fact 1 to a polygon in $CEL(P)$ of shortest perimeter to obtain our result, provided that we can easily find such a polygon.

Lemma 2: Given a polygon P, we can find a polygon in $CEL(P)$ of shortest perimeter in time $O(nlogn)$. In particular, if P is convex then we can find a shortest polygon in $CEL(P)$ in time $O(n)$.

Sketch: Let $v_0, ..., v_{n-1}$ be the vertices of P in clockwise order. For all $1 \le j \le n-1$, $i \in \{0,1\}$, $k \in \{0,1\}$, let $L_j(i,k)$ be the minimum total length of distinct diagonals and edges of P in a chain $(v_{m_1}, v_{m_2}), (v_{m_2}, v_{m_3}), ..., (v_{m_{q-1}}, v_{m_q})$ such that:

a) if $i = 0$ then $m_1 = 0$ else $m_1 = n-1$ and $m_2 = 1$,

b) if $k = 0$ then $m_q = j$ else $m_q = j+1 \; mod \; n$, $m_{q-1} = j-1 \; mod \; n$, and the diagonal $(v_{m_{q-1}}, v_{m_q})$ lies within P,

c) for all $1 \le l < q$, $m_{l+1} = m_l + 1 \; mod \; n$ or $m_{l+1} = m_l + 2 \; mod \; n$.

Note that if $j \notin \{n-1, 0\}$ or $j = n-1$ and $i + k \le 1$ then any chain satisfying (a-c) can be interpreted as a continuous part of a polygon in $CEL(P)$. Moreover, if $j = 0$ and $i = k$, then such a chain is a polygon in $CEL(P)$. *Vice versa*, any polygon in $CEL(P)$ can be interpreted as a chain satisfying (a-c) for $j = 0$ and $\{i,k\} = \{1\}$ or $\{i,k\} = \{0\}$. If we assume that $L_j(i,k)$ is set to ∞ whenever no chain satisfying (a-c) for $L_j(i,k)$ exists, then we obtain for $2 \le j < n$, $L_j(i,0) = min(L_{j-1}(i,0) + |(v_{j-1}, v_j)|, L_{j-1}(i,1))$, and if the diagonal $(v_{j-1}, v_{j+1 \; mod \; n})$ lies within P then $L_j(i,1) = L_{j-1}(i,0) + |(v_{j-1}, v_{j+1 \; mod \; n})|$, else $L_j(i,1) = \infty$. By exploiting these equalities, we can inductively find the values of $L_j(i,k)$ for $0 \le j \le n-1$, i, $k \in \{0,1\}$ and $min(L_0(0,0), L_0(1,1))$. Simultaneously, for each of these values, we can find a chain achieving it in accordance with (a-c). The chain found for $min(L_0(0,0), L_0(1,1))$ is a polygon in $CEL(P)$ of shortest perimeter. Clearly, if P is convex then we do not have to check whether particular diagonals lie within P, and the whole dynamic programming procedure takes linear time. In the non-convex case, we can use the visibility data structure of Chazelle and Guibas from [1], which enables us to find the first edge of P intersected by a diagonal drawn from one of its endpoints in time $O(logn)$. Hence, we can also find out whether a given diagonal lies within P in time $O(logn)$. The visibility data structure requires preprocessing on P which takes $O(nlogn)$ time. Since only $O(n)$ diagonals are tested for intersection with P, the whole procedure takes $O(nlogn)$ time in the non-convex case. Instead of using the sophisticated visibility data structure of Guibas and Chazelle, we could simultaneosly find all first intersections of boundary edges of P with the diagonals of P that cut off only one vertex of P [7]. Since, the number of these diagonals is $O(n)$, by Fact 1 in [11], all the first intersections can be found during a single sweep of the plane in time $O(nlogn)$. ∎

Putting everything together, we obtain:

Theorem 1: Given a convex polygon P with n vertices and perimeter of length p, we can find a triangulation of P of length $min(2M(P), p)(\lceil log_2(n-2)\rceil + 1)$ in time $O(n)$.
Proof: Find a shortest polygon S in $CEL(P)$ by using the procedure from Lemma 1. Run the pairing, triangulation procedure from Fact 1 on S. By Lemma 1 and Fact 1, the resulting triangulation of S plus S forms a triangulation of P satisfying the thesis requirements. By Lemma 2 and Fact 1, the triangulation of P can be produced in linear time. ∎

An obvious way to generalize Theorem 1 to include non-convex polygons P is to partition P or a shortest polygon in $CEL(P)$ into convex polygons by drawing some diagonals and to triangulate each of the convex polygons according with Theorem 1. Unfortunately, no good, fast heuristic for minimum weight (i.e. length) convex diagonal partition of a polygon is presently known. The dynamic programming procedure of Greene [6] finding such an optimal convex partition runs in time $O(n^2r^2)$ where r is the number of reflex angles of P. For the time complexity reason, we may prefer to use a non-optimal, fast method of canceling concave vertices. This faster method unfortunately multiplies the logarithmic approximation factor from Theorem 1 by r. Thus, whichever way of convex diagonal partition we choose between the two mentioned, the resulting generalization of Theorem 1 is interesting only for values of r that are relatively small with comparison to n. The two possible generalizations are presented in the following theorem.

Theorem 2: Given a polygon P with n vertices, r reflex angles and the perimeter of length p, we can find a triangulation of P within an $O(logn)$ factor from $M(P)$ in time $O(n^2r^2)$, and a triangulation of P within an $O(rlogn)$ factor from the minimum of $M(P)$ and p in time $O(nlogn)$.
Proof: We begin from the proof of the first part. By applying the algorithm of Greene [6], we find a minimum length convex diagonal partition C of P in time $O(n^2r^2)$. Then, for each of the resulting convex polygons Q, we construct a shortest polygon U in $CEL(Q)$ by applying the procedure from Lemma 2. By Lemma 2, we can find all $U's$ in time $O(n)$. For the sake of proof, let us consider a shortest polygon S in $CEL(P)$. Clearly, the perimeter of U is not longer than the length of the diagonals from C on the boundary of Q plus the length of the edges of S within Q. Hence, the sum of the lengths of the perimeters of the constructed shortest polygons U is not greater than $2 \mid C \mid$ plus $\mid S \mid$. Since $\mid C \mid \leq M(P)$ and, by Lemma 1, $\mid S \mid \leq 2M(P)$, this sum is $O(M(P))$. So, if we now triangulate each of the convex polygons U using the method of Theorem 1 and combine the resulting triangulations with C then we obtain a total triangulation of P whose length is $O(M(P)logn)$. The time performance of the whole procedure is dominated by the running time of Greene's algorithm, i.e. it is $O(n^2r^2)$.
The proof of the second part is a little different. We start from the construction of a shortest polygon S in $CEL(P)$ which takes $O(nlogn)$ time by Lemma 2. Next, we apply the approximation heuristic of Hertel and Mehlhorn for minimum number convex

diagonal partition [2] to S. The heuristic runs in time $O(nlogr)$ and draws $O(r)$ diagonals. By Lemma 1, the total length of the diagonals cannot exceed $O(r \cdot min(M(P), p))$. Hence, if triangulate the convex polygons resulting from drawing these diagonals of S using the method of Theorem 1, and combine the resulting triangulations with the convex partition, we obtain a triangulation of P of length $O(min(M(P), p)rlogn)$. ∎

FINAL REMARKS

(1) Since a shortest candidate for the external layer of a convex polygon P covers at least one vertex from every three consecutive vertices of P, it cannot have more than two thirds plus two of vertices of P. Therefore, instead of the pairing triangulation procedure from Fact 1, we could iteratively use the algorithm for finding a shortest candidate for the external layer to resulting subpolygons. After $\lceil log_{\frac{3}{2}}(n-2) \rceil + 3$ iterations, we would obtain a triangulation of P within the factor of $\lceil log_{\frac{3}{2}}(n-2) \rceil + 3$ from $M(P)$ and p. Note that the above factor is only slightly worse than this given in Theorem 1, and the iterative algorithm would also run in linear time as the pairing procedure.

(2) There exists another simple method of eliminating the perimeter factor from Fact 1, pointed out by Christos Levcopoulos [7]. Instead of finding a shortest member of $CEL(P)$, we just take a member of $CEL(P)$ that cuts off only the vertices at angles of less than say 60 degrees. P can have at most two such vertices. If they are incident to a common edge, we cut only this vertex whose elimination shortens the perimeter more. By an easy argumentation, such a member of $CEL(P)$ is of length $O(M(P))$. Then, as previously, we can run the pairing triangulation procedure on the resulting subpolygon. Levcopoulos's method can be generalized to include non-convex polygons as well as ours. The advantage of the method discussed in the paper is that it finds not only the best shortcuts but also it decreases the number of vertices at least by one third during a single perimeter scan. Hence, the pairing triangulation procedure is run on a smaller input which can give a slightly better worst case performance then in the case of Levcopoulos's method. Moreover, the notions of the external triangulation layer and of a candidate for the external triangulation layer that occur in our method seem to be of interest in their own right.

ACKNOWLEDGMENTS

I would like to thank Christos Levcopoulos for interesting discussions and valuable suggestions.

REFERENCES

[1] B. Chazelle and L.J. Guibas, *Visibility and intersection problems in plane geometry*, Proc. of the 1st ACM Symposium on Computational Geometry, Baltimore.

[2] S. Hertel and K. Mehlhorn, *Fast triangulation of simple polygons*, Proc. of the International FCT-Conference, Borgholm, Sweden.

[3] P.D. Gilbert, *New Results in Planar Triangulations*, M.S. Thesis, Coordinated Science Laboratory, University of Illinois, Urbana, Illinois.

[4] D.G. Kirkpatrick, *A Note on Delauney and Optimal Triangulations*, Information Processing Letters, Vol. 10, No. 3, pp. 127-131.

[5] G.T. Klincsek, *Minimal Triangulations of polygonal domains*, Ann. Discrete Math., vol. 9, pp. 121-123.

[6] J.M. Keil and J.R. Sack, *Minimum Decompositions of Polygonal Objects*, SCS-TR-42, Carleton University, Ottawa.

[7] C. Levcopoulos, *personal communication.*

[8] A. Lingas, *Advances in Minimum Weight Triangulation*, Ph.D. dissertation, Linköping University.

[9] A. Lingas, *Greedy Heuristics for Minimum Weight Triangulation of Convex Polygons*, in preparation.

[10] E.L. Lloyd, *On Triangulations of a Set of Points in the Plane*, Proc. of the 18th Annual IEEE Conference on the Foundations of Computer Science, Providence.

[11] C. Levcopoulos and A. Lingas, *Bounds on the Length of Convex Partitions of Polygons*, Proc. of the 4th FST-TCS conference, Lecture Notes in Compt. Sci. 181, Springer Verlag, Bangalore 1984.

[12] D.T. Lee and F.P. Preparata, *Computational Geometry- A Survey*, IEEE Transactions on Computers, vol. c-33, no 12.

[13] G.K. Manacher, and A.L. Zorbrist, *Neither the greedy nor the Delauney triangulation of a planar point set approximates the optimal triangulation*, Information Processing Letters, Vol.9, No. 1, pp. 31-34.

[14] G.K. Manacher, and A.L. Zorbrist, *The use of probabilistic methods and of heaps for fast-average-case, space-optimal greedy algorithms (extended abstract)*, manuscript, 1982.

FINDING THE LARGEST RECTANGLE IN AN ORTHOGONAL POLYGON

MICHAEL MCKENNA
JOSEPH O'ROURKE
SUBHASH SURI
Department of Electrical Engineering and Computer Science,
The Johns Hopkins University, Baltimore, Maryland 21218, USA

ABSTRACT

We describe an $O(n \log^5 n)$ time $O(n)$ space algorithm for finding the largest-area rectangle that will fit in an orthogonal polygon having holes. This algorithm also allows us to find the largest perimeter rectangle in $O(n \log^4 n)$ time and $O(n)$ space. Our algorithm uses less space and is simpler than an already-existing algorithm that solves a similar problem in $O(n \log^3 n)$ time and $O(n \log n)$ space.

1. INTRODUCTION

We show an $O(n \log^5 n)$ time $O(n)$ space algorithm for finding the largest-area untilted rectangle in an orthogonal polygon P having holes, and we show an $O(n \log^4 n)$ time $O(n)$ space algorithm for finding the largest-perimeter rectangle in P.

The problem of finding the largest-area or largest-perimeter rectangle in orthogonal polygon P is a generalization of the problem in [1], where the largest-area empty rectangle r in a rectangle R having interior points is found in $O(n \log^3 n)$ time $O(n \log n)$ space. Their approach can be used to find the largest-perimeter empty rectangle r' in the same time and space bounds. It turns out that our algorithm can be used to solve the problem in [1], and their algorithm can be used to solve our problem. Our algorithm has value in that it uses less space and it is simpler. For example, it requires no data structure more complicated than a balanced tree.

In this discussion, "largest rectangle" will refer to both the largest-area and largest-perimeter rectangles. Every procedure demonstrated will require only $O(n)$ space.

2. A DIVIDE-AND-CONQUER APPROACH

Let P be an orthogonal polygon with orthogonal holes in general position; that is, no two edges of P are on the same line. The generality of position is assumed only to simplify this discussion. Without loss of generality, assume that half the vertices of P are to the left of the y-axis. At the highest level our algorithm is as follows:

(1) Find the largest rectangle in P that is to the left of the y-axis, using 3 steps:
  (1.1) Collect the boundaries of P that are on the left of the y-axis.
  (1.2) Modify these boundaries to create a polygon P_l within which a largest rectangle on the left would lie. (See the shaded region in Figure 1.) If the largest rectangle in P is to the left of the y-axis, then this largest rectangle will not touch the y-axis; so P_l can be created by redefining some of the vertices of P that are near the y-axis. (In Figure 1, the vertices labelled v have been redefined as v'.) We thus avoid adding new vertices to the problem. Note that P_l may be disconnected.
  (1.3) Find the largest rectangle in P_l.
(2) In a way similar to step 1, find the largest rectangle in P that is to the right of the y-axis. This involves the creation of a polygon P_r lying on the right of the y-axis. (See Figure 1.)
(3) Find the largest rectangle in P that crosses or touches the y-axis, using two steps:
  (3.1) Outline the *axis-region* R of polygon P that is horizontally visible from the y-axis. (See the shaded region in Figure 2.) Note again that R might consist of several disconnected parts R_1, R_2,...
  (3.2) Find the largest rectangle in each part R_i of the axis-region.

By sorting the horizontal edges of P that cross the y-axis, we can determine the boundaries of P_l and P_r in $O(n \log n)$ time; and by sweeping vertical lines leftward and rightward from the y-axis, we can determine the boundary of axis-region R in $O(n \log n)$ time. However, it is the execution of step 3.2 that dominates this divide-and-conquer algorithm. We claim that the largest-area rectangle in the axis-region can be found in $O(n \log^4 n)$ time and that the largest-perimeter rectangle in the axis-region can be found in $O(n \log^3 n)$ time. Thus the largest-area rectangle in polygon P will be found in time

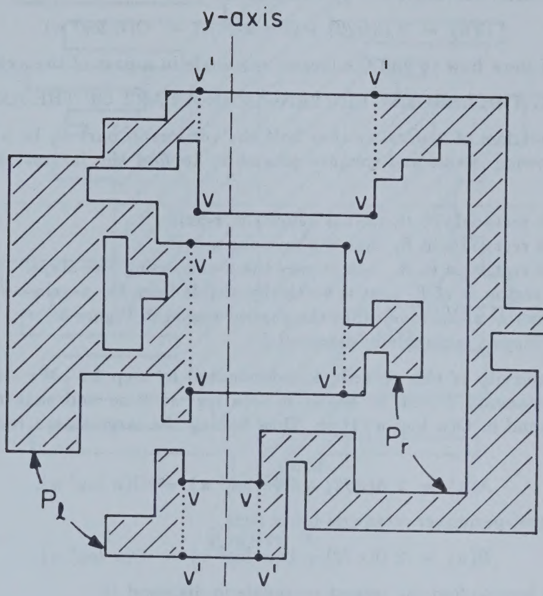

y-axis
v'
v'
v
v
v'
v
v
v'
P_r
P_l
v
v
v'
v'

Figure 1

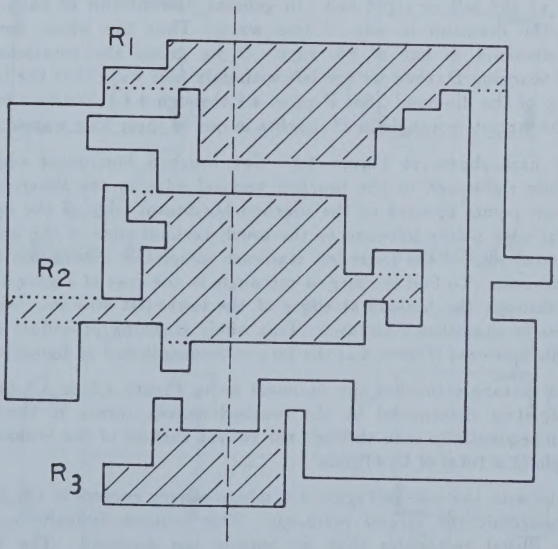


Figure 2

$$A(n) = 2\,A(n/2) + O(n \log^4 n) = O(n \log^5 n),$$

and the largest-perimeter rectangle will be found in time

$$B(n) = 2\,B(n/2) + O(n \log^3 n) = O(n \log^4 n).$$

We must then show how to find the largest rectangle in a part of the axis-region.

3. FINDING THE LARGEST RECTANGLE IN A PART OF THE AXIS-REGION

Assume without loss of generality that half the vertices in part R_i lie above the x-axis. We can apply the following divide-and-conquer procedure to find the largest rectangle in R_i: (See Figure 3.)

(1) Find the largest rectangle in R_i that is below the x-axis.
(2) Find the largest rectangle in R_i that is above the x-axis.
(3) Find the largest rectangle in R_i that crosses the x-axis, using two steps:
    (3.1) Find the region D of R_i that is vertically visible from the x-axis.
        Call region D a *diamond.* (See the shaded region in Figure 3.)
    (3.2) Find the largest rectangle in diamond D.

The time complexity of this procedure is dominated by step 3.2. We claim that the largest-area rectangle in diamond D can be found in $O(n \log^3 n)$ time and that the largest-perimeter rectangle can be found in $O(n \log^2 n)$ time. Thus finding the largest-area rectangle in axis-region R takes time

$$A(n) = 2\,A(n/2) + O(n \log^3 n) = O(n \log^4 n),$$

and finding the largest-perimeter rectangle takes time

$$B(n) = 2\,B(n/2) + O(n \log^2 n) = O(n \log^3 n).$$

We now show how to find the largest rectangle in diamond D.

4. FINDING THE LARGEST RECTANGLE IN A DIAMOND

Note that the interior of the top edge of the largest rectangle must touch the boundary of diamond D either at the left or right end. In general, the interior of each edge of the largest rectangle touches the diamond in one of two ways. Thus the whole rectangle touches the boundary of the diamond in one of 16 ways. If we group the rotationally and reflexively symmetric ways of touching D, then we are left with only four ways that the largest rectangle can touch the boundary of the diamond. (See Figures 4.1 through 4.4.) Without loss of generality, we will assume that the largest rectangle in D touches in one of these four ways.

Consider the case shown in Figure 4.1. The touched horizontal edge in the lower-left *staircase* of D points rightward to the touched vertical edge in the lower-right staircase. This vertical edge in turn points upward to the touched horizontal edge of the upper-right staircase; and that horizontal edge points leftward to the touch vertical edge of the upper-left stair. Thus the touched horizontal edge of the lower-left staircase completely determines the touched edges in the three other staircases. To find the largest rectangle in the case of Figure 4.1, it then suffices to sequentially scan through the horizontal edges of the lower-left staircase while maintaining the corresponding edges in the other staircases. This whole scanning procedure can be performed in $O(n)$ time; so in this first case (Figure 4.1) the largest rectangle can be found in $O(n)$ time.

If the largest rectangle touches the diamond as in Figure 4.2 or 4.3 then the edges of the rectangle are completely determined by the touched convex corner in the lower-left staircase. Here again, we can sequentially scan through the convex corners of the lower-left staircase to find the largest rectangle in a total of $O(n)$ time.

That leaves us with the case in Figure 4.4, where convex corners in the lower-left and upper-right staircases determine the largest rectangle. This is more difficult, because some pairs of corners determine illegal rectangles that go outside the diamond. The time complexity for handling this fourth case is more than linear, so the time-complexity for finding the largest rectangle in a diamond will be determined by the time to handle the fourth case.

For a given convex corner c in the lower-left staircase of diamond D, let top(c) be the horizontal edge of the upper-left staircase that is encountered when a vertical line is drawn

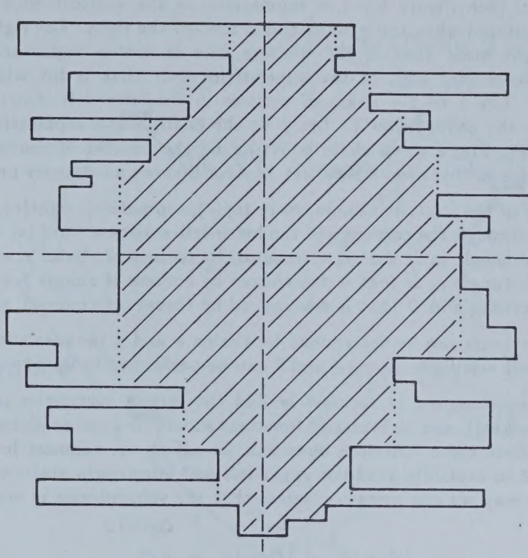


Figure 3

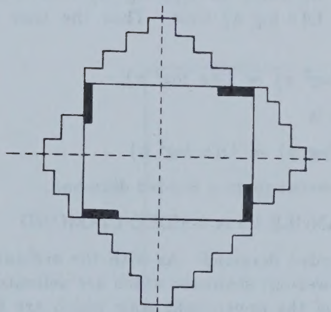


Figure 4.1

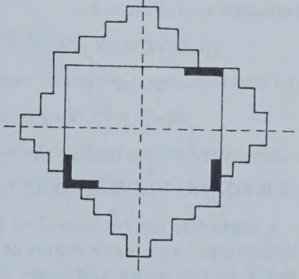


Figure 4.2

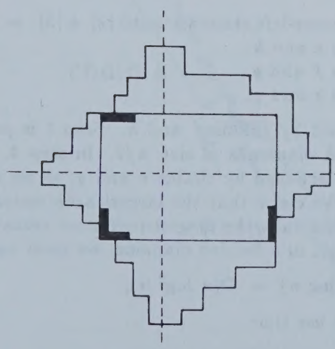


Figure 4.3

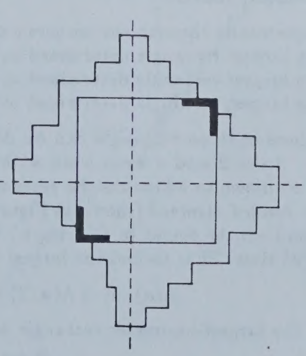


Figure 4.4

upward from corner c. (See Figure 5.) Let topmate(c) be the vertical edge of the upper-right staircase that is encountered when top(c) is extended toward the right. Let right(c) be the vertical edge of the lower-right stair that is hit when a line is drawn rightward from c, and let rightmate(c) be the horizontal edge of the upper-right stair that is hit when a line is drawn upward from right(c). Let e be the chain of vertices in the lower-left staircase that are above corner c, and let f be the chain below c. Let g be the chain above topmate(c), and let h be the chain below topmate(c). For a given chain α, let $|\alpha|$ be the number of convex corners in α. To find the largest rectangle in this case, we use yet another divide-and-conquer procedure:

(1) For every corner c of the lower-left chain, store top(c), topmate(c), right(c), and rightmate(c).
(2) Scan sequentially through the corners c of the lower-left staircase until $|e| + |h| = |f| + |g|$.
(3) Find the largest rectangle in D that is determined by corners of chains e and h.
(4) Find the largest rectangle in D that is determined by corners of chains f and g.
(5) Find the largest rectangle in D that is determined by corners of chains f and h.

Note that no rectangle can be determined by chains e and g because such a rectangle would go beyond the upper-left staircase. Steps 1 and 2 can be performed in linear time.

Steps 3 and 4 are assigned to recursively find the largest rectangles in smaller diamonds. The top(), topmate(), right(), and rightmate() functions should be kept as global arrays of pointers, so that we may calculate these functions once and for all at the topmost level of the recursion. We then have no need to explicitly send the upper-left and lower-right stairs to the lower levels of the recursion. In this way we can assure ourselves that the subproblems in steps 3 and 4 are each of size

$$|e| + |h| = |f| + |g| = n/2.$$

In step 5, the rectangles determined by chains f and h are not constrained by any upper-left chain. We are thus here working with a *3-sided diamond* (shown in Figure 6). We claim that the largest-area rectangle in a 3-sided diamond can be found in $O(n \log^2 n)$ time, and that the largest-perimeter rectangle here can be found in $O(n \log n)$ time. Thus the time to find the largest-area rectangle in a diamond is

$$A(n) = 2\,A(n/2) + O(n \log^2 n) = O(n \log^3 n),$$

and the time to find the largest-perimeter rectangle is

$$B(n) = 2\,B(n/2) + O(n \log n) = O(n \log^2 n).$$

We now show our procedure for finding the largest rectangle in a 3-sided diamond.

5. FINDING THE LARGEST RECTANGLE IN A 3-SIDED DIAMOND

Let c be a lower-left convex corner of the 3-sided diamond. As with the ordinary diamond, let e and f be the upper and lower chains of the lower-left staircase which are delimited by corner c, but let g and h be the upper and lower chains of the upper-right stair which are delimited by rightmate(c). (See Figure 6.) To find the largest rectangle in this 3-sided diamond, we use another divide-and-conquer routine:

(1) Scan sequentially through the corners c of the lower-left staircase until $|e| + |h| = |f| + |g|$.
(2) Find the largest rectangle determined by chains e and h.
(3) Find the largest rectangle determined by chains f and g.
(4) Find the largest rectangle determined by chains e and g.

Note here that no rectangle can be determined by chains f and h. Step 1 is performed in linear time. Steps 2 and 3 each work with 3-sided diamonds of size $n/2$. In step 4, there is no lower-right staircase to constrain the rectangles determined by chains e and g, so we are working here with a *2-sided diamond* (shown in Figure 7). We claim that the largest-area rectangle in a 2-sided diamond can be found in $O(n \log n)$ time, and that the largest-perimeter rectangle can be found in $O(n)$ time. Thus to find the largest rectangle in a 3-sided diamond, we must use time

$$A(n) = 2\,A(n/2) + O(n \log n) = O(n \log^2 n),$$

and to find the largest-perimeter rectangle we must use time

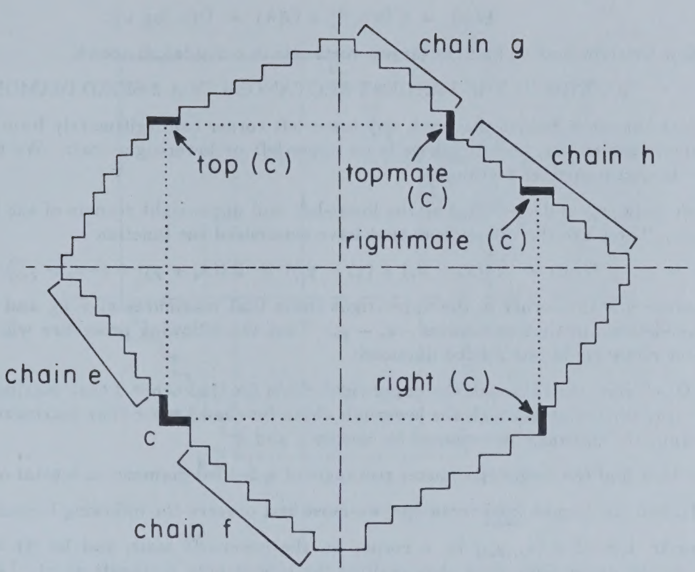

chain g
top (c)
topmate (c)
chain h
rightmate (c)
chain e
right (c)
c
chain f

Figure 5

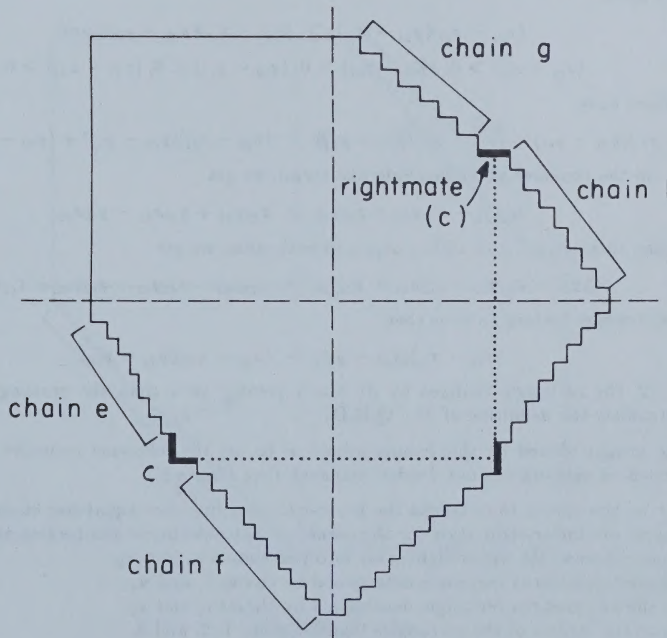

chain g
rightmate (c)
chain h
chain e
c
chain f

Figure 6

$$B(n) = 2\,B(n/2) + O(n) = O(n \log n).$$

Now we show how to find the largest rectangle in a 2-sided diamond.

6. FINDING THE LARGEST RECTANGLE IN A 2-SIDED DIAMOND

Note that in a 2-sided diamond, any lower-left corner can legitimately form a rectangle with any upper-right corner, because there is no upper-left or lower-right stair. We first show how to find the largest-perimeter rectangle.

Let $c=(x_c,y_c)$ and $d=(x_d,y_d)$ be the lower-left and upper-right corners of the largest perimeter rectangle. These are the two corners that have maximized the function

$$f(c,d) = 2\,((x_d - x_c) + (y_d - y_c)) = 2\,((x_d + y_d) + (-x_c - y_c)).$$

Thus corner d is the corner in the upper-right chain that maximizes $x_d + y_d$, and c is the corner in the lower-left chain that maximizes $-x_c - y_c$. Thus the following procedure will find the largest-perimeter rectangle in our 2-sided diamond:

(1) In $O(n)$ time, scan through the upper-right chain for the corner d that maximizes $x_d + y_d$.
(2) In $O(n)$ time, scan through the lower-left chain for the corner c that maximizes $-x_c - y_c$.
(3) Output the rectangle determined by corners c and d.

We can thus find the largest-perimeter rectangle of a 2-sided diamond in a total of $O(n)$ time.

To find the largest area rectangle, we must first observe the following lemma:

Lemma 1: Let $c1 = (x_{c1},y_{c1})$ be a corner on the lower-left stair, and let $d1 = (x_{d1},y_{d1})$ be the corner on the upper-right stair that realizes the largest-area rectangle at c1. Let c2 be a corner lower than c1 in the lower-left stair, and let d2 be the corner of the upper-right stair that realizes the largest-area rectangle at c2. Corner d2 is no higher than corner d1 in the upper-right stair.

Proof: (See Figure 7.) For contradiction, assume that d2 is higher than d1 in the upper-right stair. We have

$$(x_{d1} - x_{c1})(y_{d1} - y_{c1}) \geq (x_{d2} - x_{c1})(y_{d2} - y_{c1}), \text{ and}$$
$$(x_{c2} - x_{c1}) > 0,\ (y_{d2} - y_{d1}) > 0,\ (y_{c2} - y_{c1}) < 0,\ (x_{d1} - x_{d2}) > 0.$$

We therefore have

$$(x_{d1} - x_{c1})(y_{d1} - y_{c1}) + (x_{c2} - x_{c1})(y_{d2} - y_{d1}) > (x_{d2} - x_{c1})(y_{d2} - y_{c1}) + (y_{c2} - y_{c1})(x_{d1} - x_{d2}).$$

Breaking up the products and cancelling like terms, we get

$$x_{d1}y_{d1} + x_{c2}y_{d2} - x_{c2}y_{d1} > x_{d2}y_{d2} + y_{c2}x_{d1} - y_{c2}x_{d2}.$$

Rearranging these terms, and adding $x_{c2}y_{c2}$ to both sides, we get

$$x_{d1}y_{d1} - x_{d1}y_{c2} - x_{c2}y_{d1} + x_{c2}y_{c2} > x_{d2}y_{d2} - x_{d2}y_{c2} - x_{c2}y_{d2} + x_{c2}y_{c2}.$$

Grouping common factors gives us then

$$(x_{d1} - x_{c2})(y_{d1} - y_{c2}) > (x_{d2} - x_{c2})(y_{d2} - y_{c2}).$$

Thus at c2, the rectangle realized by d1 has a greater area than the rectangle realized by d2. This contradicts the definition of d2. **Q.E.D.**

The insight offered by this lemma allows us to use the following recursive procedure to find the largest-area rectangle in our 2-sided diamond: (See Figure 8.)

(1) Let c be the corner that breaks the lower-left stair into two equal-size chains l_1 and l_2. Scan through the upper-right stair for the corner d that determines the largest-area rectangle at c. Corner d breaks the upper-right stair into two chains u_1 and u_2.
(2) Find the largest-area rectangle determined by chains l_1 and u_1.
(3) Find the largest-area rectangle determined by chains l_2 and u_2.
(4) Output the largest of the rectangles found in steps 1, 2, and 3.

Lemma 1 assures us that the largest rectangle will be determined by two corners above c and d, or by two corners below c and d. Steps 2 and 3 are applied to smaller subproblems where the lower-left stair is recursively halved; so the recursion goes $O(\log n)$ levels deep. At any one level

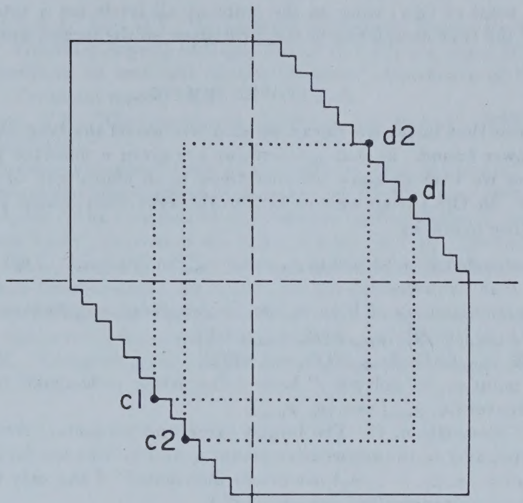


Figure 7

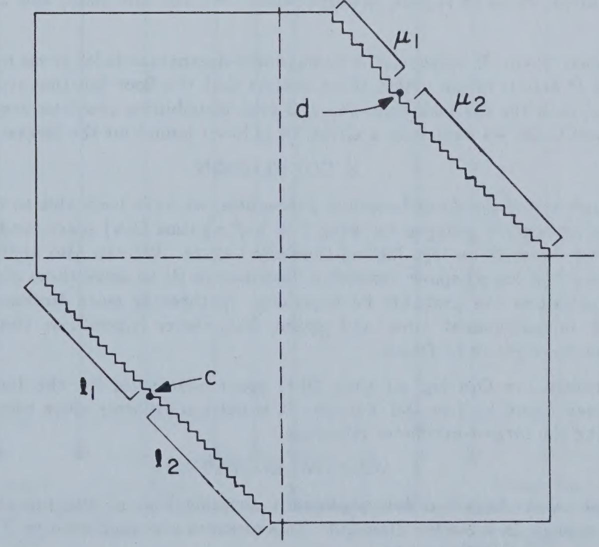

μ_1
d
μ_2
c
ℓ_1
ℓ_2

Figure 8

of the recursion, the scans of the upper-right chain do not overlap; so all the scans at one level of the recursion use a total of $O(n)$ time; so the scans at all levels use a total of $O(n \log n)$ time. The scans dominate the time complexity of this procedure; so the largest-area rectangle is found in $O(n \log n)$ time.

7. A LOWER BOUND

In [5], it is shown that under the linear decision tree model the even distribution problem has an $\Omega(n\log n)$ time lower bound. In that problem, we are given n unsorted points $\{p_1, p_2, \ldots, p_n\}$ on the real line, and we wish to know whether there is an empty gap of length greater than 1 among these points. In $O(n)$ time we can reduce the even distribution problem to the largest rectangle problem: (See figure 9.)

(1) Find the leftmost and rightmost points $p_{\min}$ and $p_{\max}$ in $\{p_1, p_2, \ldots, p_n\}$.
(2) Create polygon P in two steps:
  (2.1) Let the outer boundary of P be defined by vertices at coordinates
    $(p_{\min}-1/2,\ p_{\max}+1/2)$, $(p_{\max}+1/2,\ p_{\max}+1/2)$,
    $(p_{\min}-1/2,\ p_{\min}-1/2)$, $(p_{\max}+1/2,\ p_{\min}-1/2)$.
  (2.2) For each point p_i, let polygon P have a degenerate rectangular hole defined by vertices at coordinates $(p_i,\ p_{\min})$ and $(p_i,\ p_{\max})$.
(3) Find the largest rectangle in P. The largest (area and perimeter) rectangle will be between the holes corresponding to the consecutive points p_i and p_j that are farthest apart.
(4) Output "The points $\{p_1, p_2, \ldots, p_n\}$ are evenly distributed" if and only if (area of the largest rectangle)$/(p_{\max}+1-p_{\min}) \leq 1$.

Steps 1, 2, 4 take $O(n)$ time, but the whole procedure must take $\Omega(n\log n)$ time; so finding the largest rectangle (step 3) must take $\Omega(n\log n)$ time. This reduction is similar to a reduction in [3] where the even distribution problem is reduced to the maximum gap problem, and it is similar and to a reduction in [4] where integer sorting is reduced to quadrilaterization.

The holes in P are infinitely thin because we want to avoid overlapping holes. To construct non-degenerate "thick" holes, we would have to find an appropriately small thickness; but finding this small thickness seems to require $\Omega(n\log n)$ time. We can not afford this time in our problem reduction.

Two caveats, then: If polygon P is to have non-degenerate holes or no holes at all, then our construction of P here is illegal. Also, if we assume that the floor function requires only constant time to execute, then the the maximum gap and even distribution problems require only $O(n)$ time [2]. If either case holds, we have only a trivial $\Omega(n)$ lower bound for the largest rectangle problem.

8. CONCLUSION

Using many nested divide-and-conquer procedures, we have been able to find the largest-area rectangle in an orthogonal polygon by using $O(n \log^5 n)$ time $O(n)$ space, and we have found the largest-perimeter rectangle in $O(n \log^4 n)$ time $O(n)$ space. We can also apply the more complex $O(n \log^3 n)$ time $O(n \log n)$ space algorithm described in [1] to solve these problems. These time and space complexities can probably be improved. In three or more dimensions, these problems can be solved in polynomial time and space, but clever algorithms running in low-degree polynomial time have yet to be found.

Most recently, an $O(n \log^2 n)$ time $O(n)$ space algorithm for the largest-area rectangle problem has been found by Lee and Yap [8]. It is not immediately clear whether their approach works for finding the largest-perimeter rectangle.

ACKNOWLEDGEMENTS

We thank Alok Aggarwal for a pleasant simplification of the procedure that finds the largest-area rectangle in a 2-sided diamond. This research was supported by The National Science Foundation under grant DSR83-51468.

REFERENCES

[1] Chazelle, B., Drysdale R.L., Lee, D.T., "Computing the Largest Empty Rectangle", Proc. of the Symposium on the Theoretic Aspects of Comp. Sci., Paris, France, April 1984.

[2] Gonzalez, T., "Algorithms on sets and related problems", University of Oklahoma, Dept. of Computer Science, Technical report, 1975.

[3] Lee, D.T., Preparata, F.P., "Computational Geometry - A Survey", IEEE Trans. on Comp., Vol.33, 1984, pp. 1072-1101.

[4] Lubiw, A., "Decomposing Polygonal Regions into Convex Quadrilaterals", Proceedings of the 1st Annual Symposium on Computational Geometry, pp. 97-106, June 1985.

[5] Manber, U., Tompa, M., "The Complexity of Problems on Probabilistic, Nondeterministic and Alternating Decision Trees", Journal of the ACM, Vol.32, 1985, pp. 720-732.

[6] Naamad, A., Hsu, W.L., Lee, D.T., "On the Maximum Empty Rectangle Problem", Discrete Applied Mathematics (Netherlands), vol.8, 1984, pp 267-277.

[7] Wood, D., Yap, C.K., "Computing the Convex Skull of an Orthogonal Polygon", Proceedings of the 1st Annual Symposium on Computational Geometry, pp. 311-315, June 1985.

[8] Lee, D.T., Yap, C.K., "Computing the Largest Rectangle in a Rectilinear Polygon", manuscript in preparation.

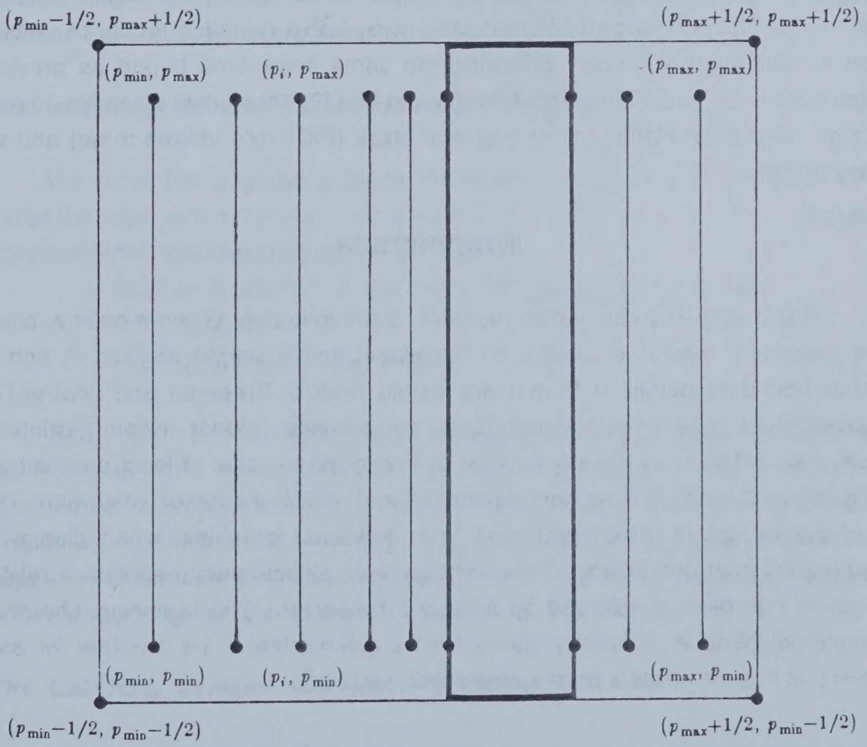


Figure 9

EFFICIENT HIDDEN-LINE ELIMINATION BY CAPTURING WINDING INFORMATION

Preliminary Version

JAMES A. DEAN
Bell-Northern Research, Dept 5K21
P.O. Box 3511
Station C
Ottawa, Ontario
K1Y 4H7

JÖRG- R. SACK[1]
School of Computer Science
Carleton University
Col. By Drive
Ottawa, Ontario
K1S 5B6

ABSTRACT

This paper introduces a tool for computational geometry, called the *labelling scheme,* for polygons. It is demonstrated that, although generated at little extra cost, the new scheme captures enough information about winding properties of polygons that simple and efficient algorithms for computing the visiblity polygon (chain) for hidden-line elimination in the perspective (parallel) model of visibility can be designed. Although exhibiting the same linear-time bound as previous algorithms such as ElGindy and Avis [6], and Lee [9], these new algorithms exhibit a high degree of clarity, utilize only one stack (ElGindy's utilizes three) and are easy to code.

INTRODUCTION

The *hidden-line elimination problem* is the following: Given a point z, called the viewpoint, which is located on the plane, and a simple polygon P, find all those boundary points of P that are visible from z. Freeman and Loutrel [7] distinguished three models of visibility for this problem: *interior* , when z is interior to P; *free exterior* , when z is exterior to P and there exists at least one half-line originating at z which does not intersect P; and *blocked exterior,* otherwise. This problem occurs in vision problems, e.g. computer graphics, when displaying two-dimensional projections of three-dimensional objects onto a screen. A related problem has been considered by Avis and Toussaint [1]: determining whether a simple polygon, P, is weakly visible from a given edge, e, i.e. whether for each point p in P there exists a point x on e which sees p.

---

[1] This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. A0332.

If one consideres the "projections" of the visible points of P onto z, the hidden-line problem can be partitioned into two cases:

parallel : z lies at infinity with respect to P; the projection lines are parallel;

perspective : otherwise.

ElGindy and Avis [6] designed a linear-time hidden-line elimination algorithm which utilizes threee stacks and turns out to be rather complex. Independently, Lee [9] and Sack [12, 13] developed single stack, linear-time algorithms (see also [11]). Lee's algorithm works for arbitrary simple polygons. While the algorithms presented in [12, 13] are simple and efficient, being based on a labelling scheme for rectilinear polygons (i.e. polygons whose edges are parallel to two given orthogonal directions), they are restricted to the rectilinear domain. Here we show how this labelling scheme can be generalized to arbitrary polygons and how efficient and simple solutions to hidden-line problems can be achieved by using this tool. For brevity all proofs are omitted and can be found in [5].

The Labelling Scheme for Rectilinear Polygons

We recall the labelling scheme for rectilinear polygons as defined in [12]: *Label the edge with minimum y-coordinate by "0"; When performing a clockwise traversal of the rectilinear polygon* $P=\{p_1,\ldots,p_n\}$*:*

if edge e_i *is labelled* l_i *and the turn at* p_{i+1} *is a right-turn then*

$l_{i+1} := l_i + 1$

$l_{i+1} := l_i - 1$ *, otherwiwse;*

See Figure 1 for an example of a rectilinear polygon labelled according to this scheme. The labelling scheme for rectilinear polygons has found several applications in problems such as hidden-line elimination, detection of structural properties, generation of rectilinear polygons, see [12, 13], and recently the convex skull problem [14]. Next we generalize the labelling to arbitrary polygons before presenting, in the next section, some of its applications.

The Labelling Scheme for Arbitrary Polygons

To extend the labelling scheme to arbitrary simple polygons, we measure the angle α made by edge e_1 and the positive horizontal originating at p_1. We perform a clockwise traversal of the edges of P and start by labelling edge e_1 with l_1, equal to the angle α divided by 100.

Subsequent labels are generated as $l_i = l_{i-1} + \theta_i /100$, where θ_i is the turn at vertex p_i, equal to 200 grads minus the interior angle at p_i. See Figure 2 for an example of a simple polygon labelled according to this scheme. Using the *gradient* as the unit of angular measure will ensure that horizontal and vertical edges have integer labels. (See [4].) To capture the complete turning behavior of the polygon we define an additional label $l_{n+1} = l_n + \theta_1/100$, see Property 3.

It is easy to observe that for any labelled simple polygon, $l_i > l_{i+1}$ iff p_{i+1} is a reflex vertex. We will call $l_1,...,l_n$ the *label-sequence* for polygon $P=\{p_1,...,p_n\}$. The label-sequence of a polygon P provides information about the winding properties of P, e.g. the larger (smaller) the label, the more polygonal chain $C_{1,i}$ spirals to the right (resp. left). We state some properties of label-sequences:

Property 1: A (not necessarily simple) polygon is convex if and only if its label-sequence is monotonically increasing.
Property 2: A polygon with monotonically increasing label-sequence is simple if and only if $l_{n+1} - l_1 = 4$.
Property 3: If a polygon P is simple then $l_{n+1} - l_1 = 4$.

Notice that the converse of Property 3 is not necessarily true. Thus the labelling sequences do not provide a linear time test for simplicity. In [4], a labelling scheme-based algorithm for testing a polygon for monotonicity has been presented. Because we choose by convention to begin labelling at edge e_1, for any given polygonal chain $C_{i,j}$ with i>j, we may relabel the chain starting at e_i so as to eliminate any ambiguities introduced by the arbitrary choice of e_1 as the starting point.

Finally, let z be some point on the plane and let v be a point on edge e_i of the boundary of polygon P. Without loss of generality, assume that line segment zv is *horizontal* and $e_i = p_i p_{i+1}$ is *down-pointing* (i.e. the y-coordinate of point z is greater than that of v). If z does not see v, then zv intersects the boundary of P somewhere else. Let e_j be the edge with the intersection closest to e_i; then e_j is up-pointing (i.e. the y-coordinate of point z is less than that of v) and is called a *blocking* edge. Further, if j>i, i.e. in the clockwise traversal the blocking edge e_j is encountered after e_i, then e_j is an *antagonist* for edge e_i. Note that the cumulative turn from an edge e_i to its antagonist e_j, represented by the difference in labels, $l_j - l_i$, is a left turn of "approximately 200 grads". See Figures 3a, 3b.

In the following section we describe two visibility algorithms. The key idea behind each is to first determine the label range of (a) the edges which are potentially visible as well as (b) their antagonists. This is done by scanning the edges of a polygon in clockwise order with respect to the viewpoint, pushing potentially visible edges on a stack and popping them off if their antagonists are encountered. All edges which do not fall into either category are ignored. The data structure for each algorithm is simply a stack containing points of the plane. When these points are connected, they define a region of visibility from the given viewpoint.

Parallel Model

Without loss of generality, we assume for the parallel model that the viewpoint is located with x-coordinate +-infinity and y-coordinate 0. Solutions for other cases can be obtained by rotating the polygon, for example. Thus the edges which are seen from the viewpoint are down-pointing. The following property is essential for obtaining the desired efficiency in the algorithms presented:

Property 4: Labels of edges visible from the viewpoint lie in the range (0, 2), while their antagonists have labels in the range (-2, 0).

We begin a traversal of the boundary of the polygon at the point with maximum y-coordinate, p_1, and terminate at the point with minimum y-coordinate, p_K. We examine edge $e_i = p_i p_{i+1}$: If its label, l_i, is in (0, 2) and has any part below the top, called top(S), of the stack , S, we push the potentially visible part , i.e. that part of e_i which is below top(S) onto S. If the label is in (-2, 0) and p_{i+1} lies to the left of the stack, then the edge is an antagonist and we pop off the part of the stack that the edge blocks. Otherwise, we continue the traversal. A pseudo-code description is given below and the interested reader is invited to follow it using the polygon in Figure 4.

Algorithm 1: Visibility from viewpoint with x = positive infinity.
Input: A simple polygon P, labelled according to the labelling scheme.
Output: The visibility chain with respect to the viewpoint.

```
BEGIN
    S := $\emptyset$;   push ($p_1$,S);
    FOR i := 1 TO n DO
     IF ($l_i \in (0,2)$) AND (Some part of $e_i$ lies below top(S)) THEN
        (* Edge $e_i$ is potentially visible from the viewpoint. *)
        (* Push the visible part of $e_i$ by finding the intersection of the horizontal *)
        (* line through top(S) and $p_i$.  Always end by pushing $p_{i+1}$. *)
    ENDIF
    ELSE IF ($l_i \in (-2,0)$) AND (y-coordinate of $p_{i+1}$ > y-coordinate of top(S)) THEN
         (* $e_i$ could be an antagonist *)
        WHILE ($p_{i+1}$ lies above top(S)) AND ($p_{i+1}$ lies to the left of the edge in S
                 containing the same y-coordinate) DO
            old_top = top(s);
            pop(s);   (* remove what antagonist blocks *)
        ENDWHILE;
         IF ($p_{i+1}$ is below top(S)) THEN
            q :=     intersection of horizontal through $p_{i+1}$ and segment (top(s),
                     old_top);
            push(q,S);
            push($p_{i+1}$,S)
        ENDIF;
    ENDFOR;
END of algorithm.
```

Theorem 1: Algorithm 1 efficiently computes the visibility of a polygon from a viewpoint with x = positive infinity in linear time.
Proof: Omitted.

Perspective Model

In this section we describe an algorithm for computing the visibility from a point z interior to a polygon P. The algorithm has a flavour similar to Lee's algorithm [9] but is based on the labelling scheme resulting in gaining the desired efficiency. To describe the algorithm, we first partition the plane into quadrants as follows: let L_{+x} and L_{-x} be the positive and negative x-axes, respectively, with origin at viewpoint z. Similarly, let L_{+y} and L_{-y} represent the positive and negative y-axes. Since $z \in P$, each ray L intersects the boundary of P in one or more points. For each half-line L we find, using Euclidean distance, the intersection point of L with the boundary of P closest to z. The four intersection points thus computed define four canonical chains; i.e. polygonal chains coincident with the boundary of P but with endpoints equal to the four intersection points, not necessarily vertices of P.

The algorithm we shall describe computes, with respect to z, the visibility chain for each canonical chain created according to the plan above. By joining the four visibility polygons, we create the visibility polygon of P with respect to z. For descriptive purposes, we concentrate on the lower right quadrant, the quadrant formed by the positive x-axis, L_{+x}, and the negative y-axis, L_{-y}, but the method is similar for the other quadrants; for example, by rotation or replacing the labels of visible edges and antagonists.

As described earlier, let L_{+x} and let L_{-y} define canonical chain $C_{i,j}$ such that u is an endpoint of $C_{i,j}$ and is the intersection of edge e_i of P with L_{+x}. Similarly, v is the intersection of edge e_j of P with L_{-y}. This is illustrated in Figure 5. We assume that P is labelled with edge e_i having label l_i. We define the *base* to be (l_i DIV 4) * 4, i.e. the greatest integer multiple of 4 less than l_i. By definition, z sees both u and v. Furthermore, we have the following:

Property 5: Edge $e_i=p_ip_{i+1}$ has label $l_1 \in$ (base,base + 2). Similarly, e_j has label $l_{j-i+1} \in$ (base + 1, base + 3).

As in Algorithm 1, the data structure is a stack containing boundary points of P. We define a ray, R, originating at Z. Initially, R = L_{+x}, and is associated with label l_R = base. R will roatate from its original position to its final position in which it is incident with L_{-y}. Since R is an oriented half-line we can speak of *to the left of* and *to the right of R*.

Initially, we push point u and point p_{i+1} on the stack, and process edges e_k for k = i+1,...,j as follows: Let R be directed through the point currently at the top of the stack. We compute θ_R, the angle formed by L_{+x} and R, and attach label l_R=base + θ_R/100 to R. The remainder of algorithm is then based on the following Lemma:

Lemma 1: (a) The current edge, e_k, examined, may be all or partially visible from z if e_k intersects R and $l_R \le l_k < l_R+2$;
(b) e_k may be an antagonist for some edge(s) already in the stack if e_k intersects R and $l_R+2 \le l_k < l_R+4$; and
(c) e_k is neither visible from z nor an antagonist, otherwise.
Proof: Omitted. An instance of each condition is depicted in Figure 6.

Using Lemma 1, we have a simple test: if e_k does not intersect the quadrant or does not intersect ray R, then we increment k and continue. Otherwise, we examine label l_k: If $l_k < l_R$, then R and e_k form a left turn and e_k is not visible from z, as illustrated in Figure 6a; thus we simply increment k and continue. If $l_R \leq l_k < l_R+$ 2, then at least part of e_k is potentially visible from z; we find the intersection of R and e_k, push it as well as p_{k+1} on the stack, increment k, and continue (See Figure 6b). If $l_R+2 \leq l_k < l_R+4$, then the edge is an antagonist; we move R until it is directed from z through p_{k+1}, pop points off the stack until the top of the stack lies on or to the left of R, push the intersection point of R and the last edge popped off the stack, push p_{k+1}, increment k and continue (Fig. 6c). Otherwise, we just increment k and continue (Fig. 6d).

In case the polygon leaves the quadrant, an test, similar to, what Lee calls a "precautinary step", in his algorithm, must be implemented. Lemma2 discusses this situation. Note that this special case is not necessary in Algorithm 1 because all the edges lie between $p_{y\text{-max}}$ and $p_{y\text{-min}}$ when viewed from x at positive infinity.

Lemma 2: (A) If e_k is intersects R with p_k on or to the right of R, and k+m is the smallest index such that a) e_{k+m} intersects R with p_{k+m} on the left of R; b) the number of intersections of the polygonal chain consisting of $e_k,\ldots,e_{k+m}$ with R is even; and c) the intersection of e_{k+m} with R is the closest of these intersections to the point at the top of the stack, then edges $e_{k+1},\ldots,e_{k+m-1}$ are not visible from z.

(B) Similarly, if e_k intersects L_{-y} with p_k on or to the left of L_{-y}, and k+m is the smallest index such that a) e_{k+m} intersects L_{-y} with p_{k+m} on the right of L_{-y}; b) the number of intersections of the polygonal chain consisting of $e_k,\ldots,e_{k+m}$ with L_{-y} is even; and c) the intersection of e_{k+m} with L_{-y} is the closest of these intersections to the point v, then edges $e_{k+1},\ldots,e_{k+m-1}$ are not visible from z.

The algorithm tests only selected edges in the quadrant, and in fact can often ignore other edges as well, illustrating the efficiency of this algorithm. We now present a pseudo-code description of the algorithm, assuming a procedure INCREMENT based upon Lemma 2.

Algorithm 2: Visibility from point z interior to polygon P.

Input: A simple polygon P and an interior viewpoint z.

Output: The visibility polygon of P with respect to z.

```
BEGIN
    find four canonical chains $C_{i,j}$ defined by the axes through z;
    FOREACH $C_{i,j}$ DO
        rotate P so that $C_{i,j}$ begins at u on $L_{+x}$ and ends at v on $L_{-y}$;   S := ø;
        push(u,S);   push($p_{i+1}$,S);   k := i + 1;
        WHILE k ≤ j DO
            direct ray R from z through top(S) and compute label $l_R$;
            IF $e_k$ intersects R THEN
                IF $l_R \leq l_k < l_R+2$ THEN  (* Edge $e_k$ is potentially visible *)
                IF top(S) <> intersection($e_k$,R) THEN push(intersection($e_k$,R),S);
                IF $e_k$ intersects $L_{-y}$ THEN push(intersection($e_k$,$L_{-y}$),S)
                                             ELSE push($p_{k+1}$,S);
                ELSE  IF $l_R+2 \le l_k < l_R+4$ THEN (* Edge $e_k$ is an antagonist *)
                   direct ray R from z through $p_{k+1}$;
                WHILE top(S) lies to R's right R DO
                   old_top = top(S);
                   pop(S);
                ENDWHILE;
                IF top(S) lies to R's right THEN
                   push(intersection(R,segment(old_top,top(S))),S);
                push($p_{k+1}$,S);
             ENDIF;
          ENDIF;
          INCREMENT;
       ENDWHILE;
       VC($C_{i,j}$) := S;  undo the rotation done at the beginning, if necessary;
    ENDFOREACH;
    join the four visibility chains to create the visibility polygon with respect to z.
END of algorithm.
```

Theorem 2: Algorithm 2 efficiently computes the visibility of a polygon from a viewpoint interior to P, in linear time.

Proof: Omitted.

CONCLUSIONS AND OPEN PROBLEMS

In this paper we have presented a new scheme for capturing winding properties of polygons. It is a generalization of the corresponding scheme for rectilinear polygons[12], as well as of the notion of sinuosity, introduced by Chazelle and Incerpi [3]. Applications of the tool demonstrated here are efficient and easy to code hidden-line elimination algorithms. These algorithms have been implemented in C on a SUN graphics workstation.

Since the rectilinear labelling scheme has found several applications (see [13], as well as [14]), we suggest, as an open problem, to find further applications for problems in not necessarily rectilinear, or isothetic, computational geometry: for example, use labels to test a polygon for simplicity or for shape classification. Perhaps it is possible to apply the theory behind Algorithms 1 and 2 towards another hidden-line elimination problem, that of the question of the visibility of a set of polygons from a given viewpoint. Furthermore, the tool might be generalized to 3-dimensional objects for solving the hidden-surface problem in a similar fashion.

REFERENCES

[1] Avis, D., Toussaint, G.T., *"An Optimal Algorithm for Determining the Visibility of a Polygon from an Edge"*, IEEE Transactions on Computing, Vol. C-30, no.12, December 1981, pp. 910-914.

[2] Carlbom, I., Paciorek, J., *"Geometric Projection and Viewing Transformations"*, Computing Surveys, Vol. 10, no. 4, 1978, pp. 465-502.

[3] Chazelle, B., Incerpi, J., *"Triangulating a Polygon by Divide-and-Conquer"*, Proceedings of the 21st Allerton Conference on Communications, Control, and Computing, Monticello, IL, October 1983, pp. 448-456.

[4] Dean, J.A., *"Structural Properties of Polygons"*, Masters Thesis, School of Computer Science, Carleton University, Ottawa, Ontario, April 1985.

[5] Dean, J.A. and J.-R. Sack, *"Efficient Hidden-line Elimination by Capturing Winding Information"*, unpublished notes, Carleton University, Ottawa, Ontario.

[6] ElGindy, H., Avis, D., *"A Linear Algorithm for Computing the Visibility Polygon from a Point"*, Journal of Algorithms Vol. 2, 1981, pp. 186-197; also Technical Report SOCS 80.10, School of Computer Science, McGill University, Montreal, Quebec, September 1980.

[7] Freeman, H., Loutrel, P.P., *"An Algorithm for the Solution of the Two-Dimensional 'Hidden-Line' Problem"*, IEEE Transactions on Electronic Computers, Vol. EC-16, no. 6, December 1967, pp.784-790.

[8] Foley, J.D., Van Dam, A., *"Fundamentals of Interactive Computer Graphics"*, Addison-Wesley, Reading, MA, 1982, pp. 267-302

[9] Lee, D.T., *"Visibility of a Simple Polygon"*, Computer Vision, Graphics, and Image Processing, Vol. 22, 1983, pp. 201-221.

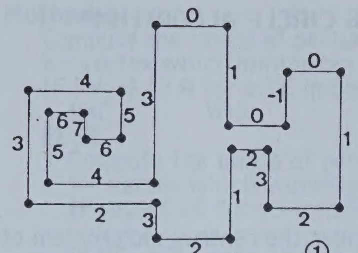
[10] Newman, W.M., Sproull, R.F., *"Principles of Interactive Computer Graphics"*, McGraw-Hill, New York, 1979, pp. 333-363.

[11] Rappaport, D., Toussaint, G.T., *"A Simple Hidden-Line Algorithm for Star-Shaped Polygons"*, Technical Report SOCS 83.23, School of Computer Science, McGill University, Montreal, Quebec, November 1983.

[12] Sack, J.-R., *"A Simple Hidden-Line Algorithm for Rectilinear Polygons"*, Proceedings of the 21st Allerton Conference on Communication, Control, and Computing, Monticello, IL, October 1983, pp. 437-446.

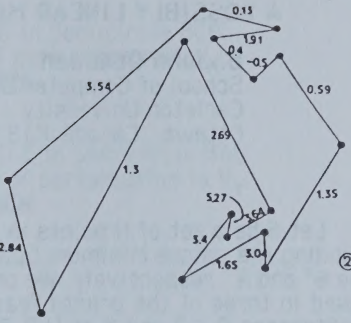
[13] Sack, J.-R., *"Rectilinear Computational Geometry"*, Ph. D. Thesis, School of Computer Science, McGill University, Montreal, Quebec, May 1984; also Technical Report SCS-TR-54, School of Computer Science, Carleton University, Ottawa, Ontario, June 1984.

[14] Wood, D., Yap, C.K, *"The Rectilinear Convex Skull Problem"*, Technical Report #132, Robotics Research and Technical Report Series, New York University, Courant Institute of Mathmatical Sciences, New York, August 1984.



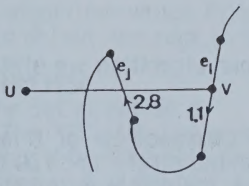
0
4
1
3
0
-1
6
7
5
0
3
5
6
1
2
4
3
1
2
2
3
2
2

①



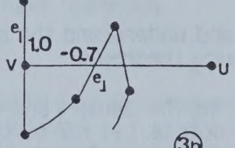
0.15
1.91
0.4
-0.5
3.54
0.59
2.69
1.3
1.35
5.27
3.64
3.4
3.04
1.65
2.84

②



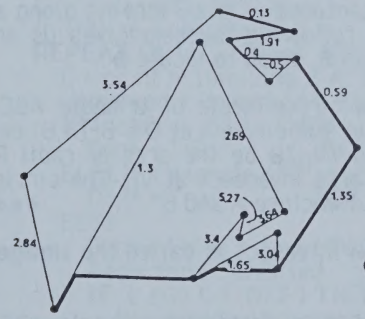
e_J
e_I
U
V
2.8
1.1

③a



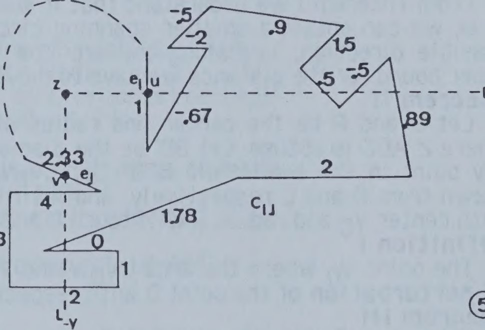
e_I
1.0
V
-0.7
U
e_J

③b



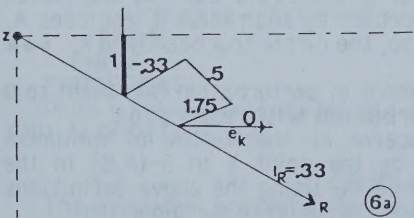
0.13
1.91
0.4
-0.5
3.54
0.59
2.69
1.3
5.27
1.35
3.64
3.4
3.04
1.65
2.84

④



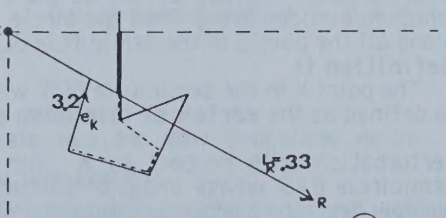
-5
.9
-2
1.5
-5
.5
Z
e_I u
L_X
1
-.67
.89
2.33
V
e_J
2
4
C_IJ
1.78
3
0
1
2
L_-Y

⑤



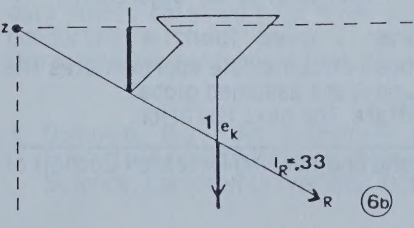
Z
1
-.33
.5
1.75
0
e_K
l_R=.33
R

⑥a

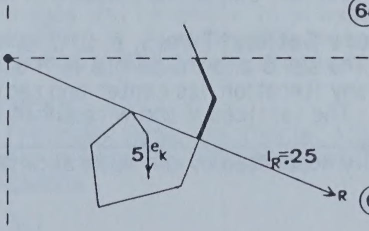


Z
3.2
e_K
l_R=.33
R

⑥c



⑥b



⑥d

A POSSIBLY LINEAR MINIMUM SPANNING CIRCLE ALGORITHM*

B. John Oommen
School of Computer Science
Carleton University
Ottawa, Canada: K1S 5B6.

INTRODUCTION

Let S be a set of N points in the plane. We consider the centruy old problem of finding the unique Minimum Spanning Circle (MSC) of S. Let its center and radius be Θ^* and R^* respectively. We present an algorithm which incorporates the ideas used in three of the primal feasible algorithms discussed in the literature. (See references 2,4,7 and 8 of [1]). Based on experimental results we conjecture that our algorithm has a **linear** time complexity.

ALGORITHM MSC

To describe and understand the mechanics of our algorithm we state without proof the following theorems.

Theorem I

Let Θ and R be the center and radius of the circumcircle of triangle ABC, where $\angle$ ABC is obtuse. Let BB' be the diameter of the circle. Let $YZ\Theta$ be the arc of radius R drawn from C. If Θ' is any point on $YZ\Theta$, the circle with center Θ' and radius R will touch the vertex C and have A and B **inside** it. ◆◆◆

From Theorem I we understand that if we are given a spanning circle centered at Θ, we can obtain a smaller spanning circle centered at Θ' by moving along a feasible direction, i.e., along the arc $YZ\Theta$. The following theorem gives us an upper bound for the distance we have to move from Θ, so as to locate Θ'.

Theorem II

Let Θ and R be the center and radius of the circumcircle of triangle ABC, where $\angle$ ABC is obtuse. Let BB' be the diameter of the circle. Let $D \in S-\{A,B\}$ be any point in the semicircle BCB'. Let UV_DW and $YV_DZ\Theta$ be the arcs of radii R drawn from D and C respectively, and let these arcs intersect at V_D. The circle with center V_D and radius R will touch C and D but enclose A and B. ◆◆◆

Definition I

The point V_D where the arcs UV_DW and $YV_DZ\Theta$ intersect is called the **image of perturbation** of the point D with respect to C.

Theorem III

Let Θ and R be the center and radius of the circumcircle of triangle ABC, where $\angle$ ABC is obtuse. Let BB' be the diameter of the circle. Let X be the vertex which minimizes $|\Theta V_D|$. Then the circle with center V_X and radius R encloses A, B and all the points in the semicircle BCB'. Also, the circle touches C and X. ◆◆◆

Definition II

The point X in the semicircle BCB' whose image of perturbation is closest to Θ is defined as the **vertex of minimum perturbation** with respect to C.

In an analogous way we can also conceive of the vertex of minimum perturbation with respect to A. This will be the point Y in S-{A,B} in the semicircle BAB' whose image of perturbation is V_Y. Using the above definitions we now describe a procedure GetNextTwo which is crucial to our algorithm.

Procedure GetNextTwo(S, A, B, C, U, V)
Input : The set S and the points A, B and C whose circumcircle approximates the MSC in any iteration. Its center and radius Θ and R are assumed global.
Output : The vertices V and W required to initiate the next iteration.

---

*Partially supported by the Natural Sciences and Engineering Research Council of Canada.

```
Method :
    Compute the image of perturbation $V_D$ for all D in Semicircle BCB'.
    X = vertex which minimizes $|\Theta V_D|$. Its image of perturbation is $V_X$.
    IF $| V_X - E | < R$ for all E in Semicircle BAB'  THEN
        V=C  ;       W=X
    ELSE
        Compute the image of perturbation $V_E$ for all E in Semicircle BAB'.
        Y = vertex which minimizes $|\Theta V_E|$. Its image of perturbation is $V_Y$.
        IF $|V_Y - D| < R$ for all D in Semicircle BCB' THEN
            V = A  ;     W = Y
        ELSE  {The case when AC is the best edge }
            V = A  ;     W = C
        ENDIF
    ENDIF
END Procedure GetNextTwo.
```

In the algorithm which follows, after each iteration, the above procedure is invoked to obtain the edge to be used in the next iteration.

ALGORITHM MSC
Input : The Set of points, S.
Output : The MSC of S. The user has the capability of deciding on the accuracy of the solution, if he does not want the exact solution.
Assumption : A procedure GetFirstTwo [1], which yields the two points to initiate the algorithm based on the Chakraborti-Chaudhuri criterion, is available.

```
Method :
    Call GetFirstTwo(S, A, B)
    Initialize  Unmarked = S  ;      Done = Satisfied = False
    REPEAT UNTIL  (Done)  OR  (Satisfied)
        C = point minimizing $\angle ADB$ for all D in Unmarked-{A,B}
        Unmarked = Unmarked - {D | $\angle ADB > \Pi/2$ }
        IF ( $\angle ACB > \Pi/2$ ) THEN
            Done = True  ;        MSC = Circle with AB as diameter
        ELSEIF (triangle ABC is acute) THEN
            Done = True  ;        MSC = Circumcircle of ABC
        ELSE
            Done = False  ;       MSC = Circumcircle of ABC
            Recompute Satisfied
            IF ( $\angle ABC > \Pi/2$ ) THEN
                Unmarked = Unmarked - {B}  ;     Call GetNextTwo (S, B, A, C, A, B)
            ELSE
                Unmarked = Unmarked - {A}  ;     Call GetNextTwo (S, A, B, C, A, B)
            ENDIF
        ENDIF
    ENDREPEAT
    Print out $\Theta$ and R of the MSC
END ALGORITHM MSC
```

CONCLUDING REMARKS

The algorithm presented here has been rigorously tested for various sets of data points obtained both from digital images and randomly. In allexperiments conducted, **without even a rare exception**, it converged in exactly two iterations. We conjecture that the algorithm has a **linear** time complexity.

REFERENCES

1. Oommen, B.J., "An Extremely Fast Minimum Spanning Circle Algorithm", Submitted for Publication. Also available as Tech. Rep. 65, School of Computer Science, Carleton University, Ottawa, Canada.

GROWING AND SHRINKING OPERATIONS ON A MONOTONE POLYGON

GERHARD ROTH
National Research Council of Canada
Division of Electrical Engineering
Ottawa, Ontario, Canada K1A OR8

ABSTRACT

Growing and shrinking operations on polygons are important techniques because they simplify spatial planning problems. Algorithms exist for convex polygons, here we describe extensions of these algorithms to enable a monotone polygon to be grown or shrunk by a convex polygon. Assume the monotone polygon has n vertices which are divided into k subchains and the convex polygon has m vertices. Then the monotone polygon can be grown or shrunk by the convex polygon in either $O(km+n)$ or $O((km+n)\log(m+n))$ time depending on the absence or presence of certain characteristics of these subchains.

INTRODUCTION

Spatial planning is an important and growing area of computational geometry with applications in robotics and VLSI layout. In general, spatial planning involves reasoning about the motion or placement of an object given some geometric constraints. The first class of problems are called findpath problems, the second class are called findspace problems. Assume that we are considering two simple polygons, one has n vertices and is fixed, the other has m vertices and can be moved. In this case the findpath problem is to find a path for the moving polygon which does not collide with the fixed polygon and the findspace problem is to find a placement of the moving polygon so that it is completely enclosed by the fixed polygon. If we do not allow the moving polygon to be rotated, which means it can only be translated, the location of any vertex completely defines the position of the polygon. For convenience vertex zero (one with the largest y coordinate) is used and is called the reference vertex. Then the findpath problem can be simplified by an operation which grows the fixed polygon to create a polygon that contains the fixed polygon. It has the characteristic that any path where the reference point of the moving polygon stays outside this polygon is a collision free path. Similarily, the findspace problem can be simplified by an operation which shrinks the fixed polygon to create a polygon contained in the fixed polygon. It has the characteristic that any placement of the moving polygon where the reference point is inside this polygon is an acceptable placement. If both polygons are convex there exist $O(m+n)$ algorithms for growing [1] and shrinking [2] the fixed polygon. Here we consider extensions of these algorithms to the case where the fixed polygon is monotone, which means that it can be partitioned into two chains whose vertices are monotone in a given direction [3].

ALGORITHM FOR GROWING AND SHRINKING

We now sketch each step of the algorithm (for more details see [4]). Assume without loss of generality that the polygon is monotone in the x direction.

Step 1: Divide the polygon into two monotone chains in $O(n)$ time.

Step 2: Both chains must be grown by the convex polygon. If the monotone polygon is to be grown the top chain is grown on the top side and the bottom chain is grown on the bottom side. If the monotone polygon is to be shrunk the sides for the growing operations are reversed on both chains. A right turn subchain consists of the maximal sequence of edges which turn right relative to their predecessor edge and a left turn subchain is defined accordingly. After a subchain has been grown or shrunk we call it an adjusted subchain. We describe in pseudo code only the growing operation on the top side of the top chain since the other cases are similar.

```
From the left most vertex to the right most vertex of the monotone chain
    Find the next subchain by collecting all the vertices of the same turn type
    Grow the right turn and shrink the left turn subchains by the convex polygon
    Connect the adjusted subchain to the current grown monotone chain
```

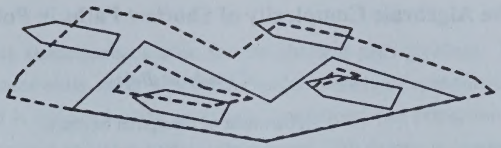


Figure 1: An example of growing and shrinking operations

A right turn subchain with r vertices can be grown by a convex polygon with m vertices using the algorithm described in [1] in $O(r+m)$ time. A convex polygon fits in a left turn subchain if at each endpoint of the subchain it can contact this point and be completely to the left of the line through each edge of the subchain. If the convex polygon fits in a left turn subchain then this subchain can be shrunk in $O(r+m)$ time using the algorithm in [2], however we use an algorithm based on supporting lines that does not require a convex hull which is of the same order. If the convex polygon does not fit in a left turn subchain we use an $O((r+m)\log(r+m))$ algorithm for the shrinking operation. Since there are k subchains and the monotone polygon has n vertices all the growing and shrinking operations are $O(km+n)$ if the convex polygon fits in all the left turn subchains and $O((km+n)\log(m+n))$ otherwise. In either case connecting the adjusted subchain to the grown chain can be done in $O(km+n)$ time because the grown chain is monotone and the moving polygon is convex.

Step 3: If the monotone polygon is to be shrunk then intersect the two grown monotone chains in $O(km+n)$ time using a modification of the slab method [5], if it is to be grown then connect the grown monotone chains in fixed time.

Theorem: *A monotone polygon with n vertices can be grown or shrunk by a convex polygon with m vertices in $O(km+n)$ time if the convex polygon fits in all the left turn subchains and $O((km+n)\log(m+n))$ time otherwise, where k is the number of subchains in the monotone polygon. If the polygon is grown the result is one polygon, if it is shrunk the result is $O(k)$ polygons. The resulting polygon(s) are monotone in the same direction as the original polygon.*

CONCLUSION

We have described an algorithm for growing and shrinking a fixed monotone polygon by a moving convex polygon which has applications in spatial reasoning. An example of this algorithm is shown in Figure 1 where the grown and shrunken polygons are drawn in dashed lines and the moving polygon is drawn in various locations. Whether this method can be extended to other classes of polygons is an open problem.

ACKNOWLEDGEMENTS

The author wishes to thank Professor Jörg Sack for his encouragement and helpful discussions.

REFERENCES

1. Lozano-Pérez, T., *Spatial Planning; Configuration Space Approach*, IEEE Transactions on Computers, February 1983.
2. Chazelle, B., *The Polygon Containment Problem*, Advances in Computing Research, Volume 1, 1983.
3. Preparata, F., Supowit, K.J., *Testing a Simple Polygon for Monotonicity*, Information Processing Letters, Vol. 12, No. 4, August 1981.
4. Roth, G., *Automatic Grasping Methods*, NRC/ERB-984, National Research Council of Canada, In Preparation.
5. Shamos, M.I., *Computational Geometry*, Ph.D Thesis, Department of Computer Science, Yale University, 1978.

The Algebraic Complexity of Shortest Paths in Polyhedral Spaces

Chanderjit Bajaj

Department of Computer Science,
Purdue University,
West Lafayette, IN 47907

Abstract: In this paper we show that the problem of finding the shortest path between two points in Euclidean 3-space, bounded by a finite collection of polyhedral obstacles, is in general not solvable by radicals over the field of rationals. The problem is shown to be not solvable even for the case when only two obstacle edges are encountered in the shortest path in 3-space. Furthermore an exact solution is shown to be not possible even under models of computation where you additionally allow the non-solvable roots of the general fifth degree polynomial equation. Hence this leaves only numerical or symbolic approximations to the shortest path solution under these models. For special relative orientations of the polyhedral obstacles however the shortest path solution is shown to be straight-edge and compass constructible. Simple polynomial time exact algorithms are known for such cases.

1. Introduction

The complexity of straight-edge and compass constructions has for long known to be equivalent to the geometric solution being expressible in terms of $(+,-,*,/,\sqrt{})$ over Q, the field of rationals [CR41],[vdW53]. In this paper we show that the problem of finding the shortest path between two points in Euclidean 3-space, bounded by a finite collection of polyhedral obstacles, is in general not solvable by *radicals*† over Q. For special relative orientations of the polyhedral obstacles however we show the shortest path to be straight-edge and compass *constructible*‡. Furthermore, in general an exact solution is shown to be not possible even under models of computation where you additionally allow the non-solvable roots of the general fifth degree polynomial equation. (Abel's now famous result was that the roots of the general fifth degree polynomial equation are not expressible in terms of radicals, [vdW53]). In our proof we generate the minimal polynomial whose root over the field of rational numbers is the solution of the shortest path problem. We show the generated polynomial to be minimal by proving it irreducible over Q and then use Galois theory to prove the above by examining the structure of the polynomial's

---

† A real number α is expressible in terms of *radicals* if there is a sequence of expressions $\beta_1, \ldots, \beta_n$, where $\beta_1 \varepsilon Q$, and each β_i is either a rational or the sum, difference, product, quotient or the k^{th} *root* of preceding βs and the last β_n is α.

‡ Henceforth by constructible we shall mean straight-edge and compass constructible.

Galois group.

A number of immediate consequences arise for the shortest path problem. First, for this problem in general there cannot exist *exact* algorithms under models of computation where the root of an algebraic equation is obtained using arithmetic operations, the extraction of k^{th} roots, as well as the extraction of the non-solvable roots of the general fifth degree polynomial equation. Second, this leaves only numerical or symbolic approximations to the optimum solution of the shortest path under the above models of computation.

2. The Shortest Path Problem

The problem of finding the shortest path between two points in Euclidean 3-space, bounded by a finite collection of polyhedral obstacles is a special case of the more general problem of planning optimal collision-free paths for a given robot system. In Euclidean 2-space (the Euclidean plane) the problem is easy to solve and the shortest path is polynomial time computable, [LW79]. The problem for Euclidean 3-space is much harder and known shortest path computations are of doubly exponential time, [SS84].

In Euclidean 3-space the shortest path between two given points, in the presence of polyhedral obstacles, can be again shown to be piecewise straight lines (*polygonal* lines), as for the planar 2-dimensional problem, with break points that lie on the edges of the given polyhedral obstacles. Since the edges of the polyhedral obstacles are arbitrary lines in Euclidean 3-space, the problem of determining the points of contact of the shortest path with these edges can without loss of generality be versed also as follows.

Given a sequence $L=(l_1,l_2,\cdots,l_n)$ *of lines in 3-dimensional space , find the shortest path from a source point* X *to a destination point* Y *constrained to pass through interior points of each of the lines* $l_1,l_2,\cdots,l_n$ *in this order.*

We identify three different cases of the relative positions of the lines. All the various configurations of the n lines in 3-space consist of combinations of these basic orientations between pairs of lines, l_i, l_{i+1}, $i=1..n-1$.

(a) Pairwise lines are parallel to each other.

(b) Pairwise lines are not parallel but intersect.

(c) Pairwise lines are skew and do not intersect.

In § 3, we show that when the lines are oriented as a combination of the cases (a) and (b), then the shortest path problem in Euclidean 3-space is constructible. Simple polynomial time exact algorithms are known for such cases, [BM85]. However in § 4, we show that for the case (c) of even two skew lines the solution is not solvable by radicals. In § 5, we extend this to the non-expressibility of the solution even with the non-solvable roots of the general fifth degree polynomial equation.

3. The Constructible Cases

We now show that when the lines are oriented as a combination of the cases (a) and (b), then the solution to the shortest path problem is always expressible in terms of $(+,-,*,/,\sqrt{\ })$ over Q.

Between pairs of lines in 3-space which are parallel to each other there exists a unique plane which contains both of them. The same applies to pairs of lines in 3-space which intersect. Also a point and a line in 3-space define a unique plane between them. The problem of finding the shortest path between X and Y in 3-space for cases (a) and (b), then reduces to a constrained 2-1/2 dimensional space problem as follows. Let the point X and line l_1 define the plane P_1, the lines l_i and l_{i+1} define the planes P_{i+1}, $i=1..n-1$, and the line l_n and the point Y define the plane P_{n+1}. The original problem is now reduced to finding the shortest path between two points X and Y in 3-space with the path constrained to the planes P_i, $i=1..n$, (Figure 1).

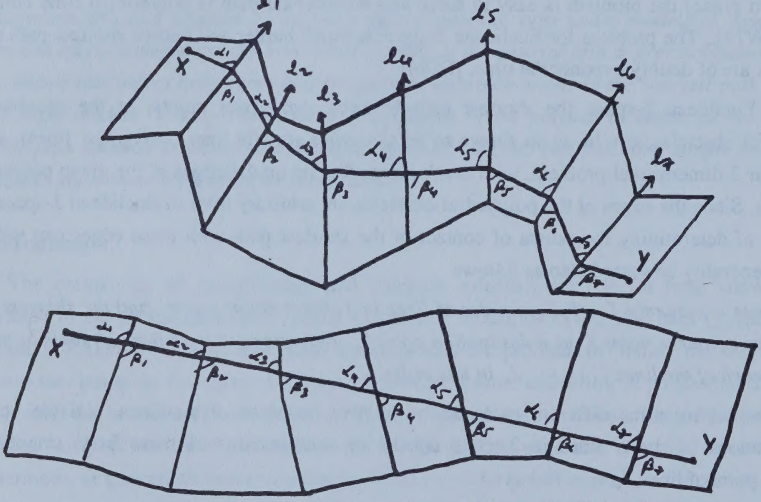

l_1
l_5
l_2
l_4
l_6
X
α_1
β_1
α_2
l_3
β_2
α_3
α_4
α_5
β_3
β_4
β_5
α_6
l_7
β_6
α_7
Y
β_7
X
α_1
α_2
α_3
β_1
β_2
β_3
α_4
α_5
β_4
α_6
β_5
β_6
α_7
Y
β_7

Figure 1: Lines which are Parallel or Intersect

On unfolding all the planes P_i to a common plane P_1, the straight line connecting X and the corresponding transformed point, Y^t is clearly the shortest path and one that subtends equal angles, at each of the lines l_i^t. The length of the path as well as the subtended angles remain invariant under the above unfoldings and thus the shortest path from X to Y which passes through the given sequence of lines l_i enters and leaves l_i at equal angles. The necessary and sufficient conditions for the shortest path can be expressed algebraically by the set of n equations

$$\alpha_i = \beta_i\ , \quad i=1..n \qquad (1)$$

where α_i and β_i are the angles subtended at line l_i by the incoming and outgoing segments of the shortest path respectively.

For the case (a) of parallel lines, since l_i is parallel to l_{i+1}, we furthermore have

$$\beta_i = \alpha_{i+1}\ , \quad i=1..n-1 \qquad (2)$$

Using the equations of (1) and (2) we obtain the equation $\alpha_1=\beta_n$ where α_1 is the angle subtended at line l_1 by the incoming shortest path segment from point X and β_i is the angle subtended at the line l_n by the outgoing shortest path segment to point Y. We are given the cartesian coordinates (a_1,a_2,a_3) and (b_1,b_2,b_3) of the points X and Y respectively and the line vectors l_i, all values over Q. Without loss of generality we can assume all lines l_i to be parallel to the x axis. The contact points on lines l_i could then be taken to be x_i,y_i,z_i where we need to determine the unknowns x_i with the values of y_i and z_i being constant. The equation $\alpha_1=\beta_n$ or equivalently the equation $Cos\,\alpha_1 = Cos\,\beta_n$ (equality of the direction cosines) is then expressed as

$$(a_1-x_1)/\sqrt{(x_1-a_1)^2+(y_1-a_2)^2+(z_1-a_3)^2} = (x_n-b_1)/\sqrt{(x_n-b_1)^2+(y_n-b_2)^2+(z_n-b_3)^2}$$

Simplifying the equation we obtain

$$x_n = b_1+(x_1-a_1)\sqrt{((y_n-b_2)^2+(z_n-b_3)^2)/((y_1-a_2)^2+(z_1-a_3)^2)}$$

The above equation has x_n expressed in terms of $(+,-,*,/,\sqrt{})$ over constant values of Q and x_1. Using sets of equations (1) and (2) it is straightforward to see that each of the unknowns x_i, $i=2..n$, can similarly be expressed in terms of $(+,-,*,/,\sqrt{})$ over constant values of Q and x_1. Next solving an equation of (1), say, $\alpha_n = \beta_n$ we obtain x_1 expressed in terms of $(+,-,*,/,\sqrt{})$ over constant values of Q. It follows that each of the x_i, $i=1..n$ is expressible in terms of $(+,-,*,/,\sqrt{})$ over constant values of Q and hence constructible.

For the case (b) of intersecting lines, since l_i intersects l_{i+1}, we have in addition to the set of equations (1) the set of equations

$$\alpha_{i+1} = \beta_i + c_i\,, \quad i=1..n-1 \qquad (3)$$

where c_i is the angle between l_i and l_{i+1} and is a constant value independent of the unknowns. This linear dependence amongst the angles is again sufficient to make each of the unknowns x_i, $i=1..n$ to be expressible in terms of $(+,-,*,/,\sqrt{})$ over constant values of Q and hence constructible.

4. The Non-Solvable Case

We now show that for the case (c) of even two skew lines the solution is not expressible by radicals. Even for skew lines the piecewise straight line shortest path enters and leaves each line l_i at equal angles [SS84]. However when any two adjacent lines l_i and l_{i+1} are skew to one another there exists no common plane containing both of them. Hence simplifications of the previous section are no longer possible. Consider the configuration of two skew lines as shown in Figure 2. We wish to obtain the shortest path between points X and Y which pass through interior points of lines l_1 and l_2 in that order.

Stating it algebraically we wish to minimize the length of the path $XC + CD + DY$ where C and D are interior points of the lines l_1, l_2.

$$minimize_{x,z}\ f(x,z) = \sqrt{(z-3)^2+5} + \sqrt{z^2+x^2+4} + \sqrt{(x-3)^2+1}$$

From the fact that in the shortest path, the path segments CD and DY subtend equal angles with the line l_2 we obtain

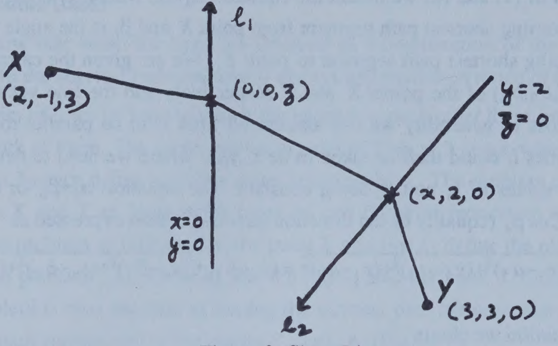


Figure 2: Skew Lines

$$x/\sqrt{x^2+z^2+4} = (x-3)/\sqrt{(x-3)^2+1}$$

and thereby

$$x = 3 + 3/(\sqrt{z^2+4}-1)$$

Substituting for x in $f(x,z)$ we reduce the shortest path problem for the above configuration to a minimization problem in a single variable.

$$\mathit{minimize}_z\ g(z) = \sqrt{(z-3)^2+5} + (1/(\sqrt{z^2+4}-1))$$
$$(\sqrt{(z^2+4)(z^2+14-2\sqrt{z^2+4}} + \sqrt{z^2+14-2\sqrt{z^2+4}})$$

The shortest path problem in Euclidean 3-space has a unique minimum, Sharir, Schorr [4]. The above function $g(z)$ can also shown to be strictly convex. Hence for the unique minimum solution the necessary and sufficient condition is $dg/dz = 0$. The corresponding rational equation is

$$dg/dz = (z-3)/\sqrt{(z-3)^2+5} + z(\sqrt{z^2+4}+1)/\sqrt{z^2+4}(\sqrt{(\sqrt{z^2+4}-1)^2+9}$$
$$-2z(\sqrt{z^2+4}+1)^2\sqrt{(\sqrt{z^2+4}-1)^2+9}/(z^2+3)^2\sqrt{z^2+4} = 0$$

By a process of repeated squaring one can eliminate all the square-roots from the rational equation above. Note that by this step we do not change the root of our original problem since repeated squaring preserves the root of the polynomial. Eliminating square-roots we obtain the polynomial,$h(z)$, with the following factorization over Q, (*Table* 1). Since the first two factors of $h(z)$ have only complex roots we obtain $q(z)$, (*Table* 1) as the polynomial whose real root is the solution to our problem. Our first step is to prove it irreducible, over Q.

Lemma 1: The polynomial $q(z)$, [*Table* 1], is irreducible over Q.

Proof : Since the monic polynomial $q(z)$ is irreducible mod the prime 47 it follows that $q(z)$ is irreducible over Q and is our minimal polynomial. ☐

To prove the non-solvability of $q(z)$ over Q by radicals we use the *Ceboratev–Van der Waerden* sampling method to determine the Galois group of $q(z)$, [Za71].

Table 1

$$
\begin{gathered}
Q: h(z)=(z^2+3)^2(z^2+4)^2(z^{20}-12z^{19}+99z^{18}-648z^{17}+3334z^{16} \\
-14976z^{15}+56370z^{14}-183360z^{13}+512669z^{12}-1255452z^{11}+2578119z^{10} \\
-5006952z^9+7175196z^8-9864504z^7+3160836z^6+26727408z^5 \\
-71592336z^4+185698656z^3-167588352z^2-25194240z+18895680) \\
Q: q(z)=z^{20}-12z^{19}+99z^{18}-648z^{17}+3334z^{16}-14976z^{15}+56370z^{14} \\
-183360z^{13}+512669z^{12}-1255452z^{11}+2578119z^{10}-5006952z^9 \\
+7175196z^8-9864504z^7+3160836z^6+26727408z^5-71592336z^4 \\
+185698656z^3-167588352z^2-25194240z+18895680 \\
Disc(q(z)): \qquad 2^i 3^j 5^k 61^l \, largeprime \ (>100) \\
Mod\ 37: \ q(z)=(z-10)(z^{19}-2z^{18}+5z^{17}-6z^{16}+18z^{15}+4z^{14}-15z^{13}+10z^{12} \\
-14z^{11}+3z^{10}-11z^9+8z^7-2z^6+17z^5-z^4-10z^3-4z^2-15) \\
Mod\ 47: \ q(z)=(z^{20}-12z^{19}+5z^{18}+10z^{17}-3z^{16}+17z^{14}-13z^{13}-7z^{12} \\
+12z^{11}-19z^{10}+5z^9-12z^8-3z^7-8z^6+12z^5-9z^4+11z^3+18z^2+16z-12) \\
Mod\ 79: \ q(z)=(z+20)(z^2-6z-15)(z^{17}-26z^{16}-34z^{15}+15z^{14}-23z^{13} \\
+2z^{12}-12z^{11}+15z^{10}-7z^9-39z^8+26z^7+2z^6+19z^5-28z^4 \\
-31z^3+33z^2-20z+9)
\end{gathered}
$$

Factorizations obtained with use of Vaxima on Unix

Lemma 2: [Za71] If $n \equiv 0 (mod 2)$ and $n > 2$ then after sampling about $(n+1)$ *good*[‡] primes we run across an $(n-1)$-cycle and an n-cycle and a permutation of the type $2+(n-3)$ and that will be enough to establish that $Gal(q(z))$ over Q is the symmetric group S_n. If $n \equiv 1 (mod 2)$ then one will run across an $(n-1)$ cycle and a permutation of the type $2+(n-2)$ in about the same time and that will be enough.

Lemma 3: [He75] $q(z) \varepsilon Q[y]$ is solvable by radicals over Q *iff* the Galois group over Q of $q(z)$, $Gal(q(z))$ is a solvable group.

Lemma 4: [He75] The symmetric group S_n is not solvable for $n \geq 5$.

Theorem 5: The shortest path problem, in general, is not solvable by radicals over Q for $n \geq 2$, where n is the number of obstacle edges encountered in the shortest path.

Proof: Restating the assertion, we need to show that the polynomial $q(z)$ of *Table* 1 is not solvable by radicals over Q. We note from *Table* 1 that for the 'good' primes $p=37,47$ and 79, the degrees of the irreducible factors of $q(z)$ mod p gives us a 2 + 17 permutation, a 20 cycle and a 19 cycle, which is enough to establish, from *Lemma* 2 for n=20, that $Gal(q(z))=S_{20}$, the symmetric group of degree 20. *Lemma* 4 tells us that this is not a solvable group and hence our assertion follows from *Lemma* 3. ☐

[‡] A good prime for a polynomial $q(z)$ is one which does not divide the discriminant of the polynomial, $disc(q(z))$.

5. Extensions to Higher Models

We now prove the non-expressibility of the shortest path solution even with the additional non-solvable roots of the general fifth degree polynomial equation. We use a part of the fundamental theorem of Galois theory which simply stated is,

> *Theorem 6: [He75]* Let $f(x)$ be a polynomial in $F[x]$, K its splitting field over F and $G(K,F)$ its Galois Group. For any subfield T of K which contains F let $G(K,T) = \{\sigma \in G(K,F) \mid \sigma(t)=t, \forall t \in T\}$. Then T is a normal extension of F iff $G(K,T)$ is a normal subgroup of $G(K,F)$ and when T is a normal extension of F, then $G(T,F)$ is isomorphic to $G(K,F) / G(K,T)$
>
> **Theorem 7:** The shortest path solution for $n \geq 2$, where n is the number of obstacle edges encountered in the shortest path, is in general not expressibile even with the additional non-solvable roots of the general fifth degree polynomial equation.

Proof : Let $q(x)$ be the minimal polynomial of *Theorem* 5, in $Q[x]$, K its splitting field over Q and $G(K,Q)$ its Galois Group. The non-solvable roots of the general fifth degree polynomial equation lie in their splitting field, say T, a normal extension of Q. The Galois group $G(T,Q)$ is, as was first proved by Abel, the symmetric group S_5, whose order is 5 !. From the Fundamental Galois Theorem 6, we obtain that $G(K,T)$ is a normal subgroup and $G(T,Q)$ is isomorphic to $G(K,Q) / G(K,T)$. However from the proof of *Theorem* 5, we know $G(K,Q)$ for the polynomial is S_{20}, of order 20 !. Further the only normal subgroups of S_{20} are the singleton set containing the identity, the Alternating group of order 20 !/2, and the symmetric group S_{20}, itself. Thus, $G(T,Q)$ cannot be isomorphic to $G(K,Q) / G(K,T)$, which goes to say that the normal extension T cannot exist and hence our assertion. ☐

Acknowledgements: I wish to thank T.T. Moh for useful discussions and Nikos Crisochoides for his help with certain calculations.

6. References

[Ba85]

Bajaj, C., *The Algebraic Complexity of Shortest Paths in Polyhedral Spaces,* Computer Science Tech. Report, Purdue University, TR85-523, 1985.

[BM85]

Bajaj, C., and Moh, T.T., *Generalized Unfoldings for Shortest Paths in Euclidean 3-Space,* Computer Science Tech. Report, Purdue University, TR85-526, 1985.

[CR41]

Courant R., and Robbins, H., *What is Mathematics ?* Oxford University Press, 1941

[He75]

Herstein, I.N., *Topics in Algebra*, 2^{nd} edition, John Wiley & sons, New York, 1975.

[LW79]

Lozano-Perez, T., and Wesley, M.A., *An algorithm for planning collision free paths among polyhedral obstacles*, CACM 22, p560-570, 1979.

[SS84]

Sharir, M., and Schorr, A., *On shortest paths in polyhedral spaces*, Proceedings 16th STOC, p144-153, 1984.

[vdW53]

van der Waerden, B.L., *Modern Algebra*, vol 1, Ungar, New York 1953.

[Za71]

Zassenhaus, H., *On the Group of an Equation* Computers in Algebra and Number Theory, SIAM and AMS proceedings, p69-88, 1971.

A ONE-DIMENSIONAL SYSTOLIC ARRAY FOR THE LARGEST EMPTY RECTANGLE PROBLEM

FRANK DEHNE
Lehrstuhl fuer Informatik I, Univ. of Wuerzburg
Am Hubland, D-8700 Wuerzburg, W.-Germany

ABSTRACT

Given a rectangle with its edges parallel to the coordinate axes containing a set S of n points in 2-dimensional Euclidean space we consider the problem of finding the largest area subrectangle with sides parallel to those of the original rectangle which contains no point of S and describe a one-dimensional systolic array which solves this problem in linear time.

1. INTRODUCTION

Given a rectangle with its edges parallel to the coordinate axes containing a set S of n points in 2-dimensional Euclidean space we consider the problem of finding the largest area subrectangle with sides parallel to those of the original rectangle which contains no point of S. [8] and [3] gave $O(n^2)$ time, linear space, and $O(n \log^3 n)$ time, $O(n \log n)$ space, respectively, algorithms to solve the problem on a sequential computer.

Motivated by [7], [11], and [2] who studied geometric problems from a similar point of view we describe a one-dimensional systolic array (see fig.1) ,called **LER**, solving the problem in linear time which is asymptotically optimal since any nontrivial computation requires time $\Omega(n)$ on a linear array.

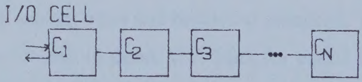


fig.1

LER will support the following operations :
- Insert/delete a point
- Report the largest empty rectangle

Since it consists of N cells, it can handle up to N points at a given time. For more details about one-dimensional systolic arrays consult [2], see also [5]. Note, that for **LER** all I/O-operations are performed by the leftmost cell C_1, since otherwise our algorithm would run in time linear with respect to N.

2. INSERTION, DELETION

To insert a new point, just put it into **LER** at its leftmost

I/O-cell and let it move to the right, until it finds an empty cell. To delete a point, send an identifier to LERs I/O-cell C_1 and let it move to the right, until it finds the specified point. Delete this point and send a signal (special record) to its right neighbor , to let the following points shift to the left and close the gap.

3. REPORTING THE LARGEST EMPTY RECTANGLE IN LINEAR TIME

3.1. Basic Structure of Algorithm

Let $S=\{s_1,\ldots,s_n\}$ be the current set of $n \leq N$ points sorted by their x-coordinates (sorting can be done in linear time applying the methods of [10] to a one-dimensional array). Let x_{min}, x_{max}, y_{min}, y_{max} be the boundaries of the bounding rectangle.

Note that each edge of the largest empty rectangle is supported by either an edge of the bounding rectangle or at least one point of S (as described in [3]). We shall call these supporting edges or points "supporting elements with resp. to S".

To simplify exposition, we shall assume, that all points of S have distinct x-coordinates and distinct y-coordinates and do not lie on the boundary. Thus, the largest empty rectangle has exactly four supporting elements with resp. to S. As we shall see at the end of this paper, the existence of some more supporting elements will not change our algorithm signifficantly.

In order to compute the largest empty rectangle, we split S into two halves $S_L=\{s_1,\ldots,s_{\lfloor n/2 \rfloor}\}$ and $S_R=\{s_{\lfloor n/2 \rfloor+1},\ldots,s_n\}$ (with their bounding rectangles adjusted) and recursively solve the problem for S_L and S_R using the systolic cells $C_1,\ldots,C_{\lfloor n/2 \rfloor}$ and $C_{\lfloor n/2 \rfloor+1},\ldots,C_n$, respectively (see fig.2).

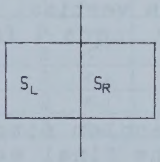


fig.2

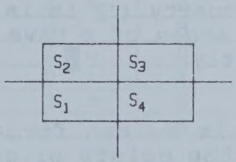


fig.3

Given the largest empty rectangles with resp. to S_L and S_R we have to compare the maximum area rectangle of these with the largest empty rectangle having at least one supporting element with resp. to S_L and S_R, respectively.

This "merging step" will be done by a second divide and conquer procedure.

After sorting S by y-coordinates, we split it into four subsets as described by fig.3 with $S_1 \cup S_2=S_L$, $S_3 \cup S_4=S_R$ and $|\ |S_2 \cup S_3| - |S_1 \cup S_4|\ | \leq 1$.

With this we recursively compute the largest empty rectangle having at least one supporting element with resp. to S_2 (S_1) and S_3 (S_4) and none with resp. to S_1 (S_2) and S_4 (S_3), repectively.

3.2. Computing the Final Result

In order to compute the final result we have to find the largest empty rectangle r, having the following property (I):

Let B_1 $(B_2,\ldots,B_4)$ be the set of supporting elements of r with resp. to S_1 $(S_2,\ldots,S_4)$, then

$|B_1|+|B_2|+|B_3|+|B_4|=4$

$|B_1|+|B_2|>0$

$|B_3|+|B_4|>0$

$|B_2|+|B_4|>0$

$|B_1|+|B_4|>0$.

Let $S_i' := S_i \cup \{p_i, q_i\}$ $(i=1,\ldots,4)$ as sketched by fig.4 .

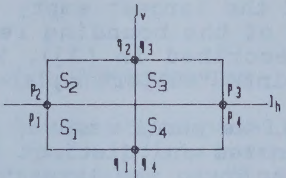


fig.4

Now, we can prove

Lemma 1:
If r is an empty rectangle with property (I) and e is an edge of r supported by a vertical (horizontal) boundary edge of S_i, then e is supported by p_i (q_i) , i=1...4 .

Proof:
From property (I) it is easy to see, that both vertical (horizontal) edges of r have to cross l_h (l_v), thus lemma 1 follows immediatly. ■

With this we can forget the bounding rectangles simply by adding the points p_i, q_i $(i=1,\ldots,4)$ during the final merging step and considering all empty rectangles with exactly four supporting points and property (I) with resp. to $S_1',\ldots,S_4'$. Note, that we add at most 4n points simultaneously, thus every cell of **LER** has to store at most 5 points.

Definition 1:

Let (x_1,y_1), (x_2,y_2) be two points, then

$(x_1,y_1) <_{ur} (x_2,y_2)$:<=> $x_1<x_2$ and $y_1<y_2$

$(x_1,y_1) <_{ul} (x_2,y_2)$:<=> $x_1>x_2$ and $y_1<y_2$

$(x_1,y_1) <_{ll} (x_2,y_2)$:<=> $x_1>x_2$ and $y_1>y_2$

$(x_1,y_1) <_{lr} (x_2,y_2)$:<=> $x_1<x_2$ and $y_1>y_2$

Let M be a set of points and $x \in M$, then x is called a ur-maximal [ul-maximal, ll-maximal, lr-maximal] element of M :<=> x is a maximal element of M with resp. to $<_{ur}$ [$<_{ul}$, $<_{ll}$, $<_{lr}$].

Let M_1 $[M_2,M_3,M_4]$ be the ur [lr,ll,ul] -maximal elements of S'_1 $[S'_2,S'_3,S'_4]$, then we have

Lemma 2:
Let r be an empty rectangle supported by four points $\{t_1,\ldots,t_4\}\subseteq S'_1\cup S'_2\cup S'_3\cup S'_4$ with property (I), then $\{t_1,\ldots,t_4\}\subseteq M_1\cup M_2\cup M_3\cup M_4$.

Proof:
From property (I) it easy to see, that both vertical (horizontal) edges of r cross l_h (l_v). Since r has to be empty lemma 2 follows immediatly. ■

Summarizing this, we have

Theorem 1:
In order to compute the final result it is sufficient to find the maximum area rectangle of all empty rectangles supported by four points $\{t_1,\ldots,t_4\}\subseteq B_1\cup B_2\cup B_3\cup B_4$ with $B_i\subseteq M_i$ $(i=1,\ldots,4)$, $|B_1|+|B_2|+|B_3|+|B_4|=4$, $|B_1|+|B_2|>0$, $|B_3|+|B_4|>0$, $|B_2|+|B_3|>0$ and $|B_1|+|B_4|>0$.

CASE	|B1|	|B2|	|B3|	|B4|	TYPE
1)	0	1 <	1 ^	2 ∨>	B
2)	0	1 <	2 ^>	1 ∨	B
3)	0	2 <^	0	2 ∨>	A
4)	0	2 <^	1 >	1 ∨	B
5)	1 <	0	1 ^	2 ∨>	B
6)	1 <	0	2 ^>	1 ∨	B
7)	1 <	1 ^	0	2 ∨>	B
8)	1 <	1 ^	1 >	1 ∨	C
9)	1 ∨	1 <	1 ^	1 >	C
10)	1 ∨	1 <	2 ^>	0	B
11)	1 ∨	2 <^	0	1 >	B
12)	1 ∨	2 <^	1 >	0	B
13)	2 <^	0	1 >	1 ∨	B
14)	2 <∨	0	2 ^>	0	A
15)	2 <∨	1 ^	0	1 >	B
16)	2 <∨	1 ^	1 >	0	B

table 1

In table 1 all possible (16) cases are listed with < [>,^,∨] denoting that a point supports a left [right, upper, lower] edge of an empty rectangle.
There are essentially three types of empty rectangles which we have to consider.
A type A rectangle is supported by two points of M_1 $[M_2]$ and M_3 $[M_4]$, respectively.
A type B rectangle is supported by two points of one quadrant and one point each of two other quadrants, while a type C rectangle is supported by one point of each quadrant.
It is easy to see that the directions of support as given in table 1 are the only possible ones:

For both cases of type A rectangles this is trivial. Concerning type B rectangles let's for exemple look at case

5. There is only one supporting point in the left half, which must be a left support, since otherwise the rules of theorem 1 would be violated. There is only one supporting point in the upper quadrant of the right half, which must be an upper support for the same reason.
   Having exactly one supporting point in each quadrant (type C), there are two possible cases. The supporting point sp_1 in the lower left quadrant M_1 must either be a left or lower support, since otherwise there would be no supporting point in M_2 or M_4, respectively. Assuming sp_1 to be a left [lower] support, it is easy to see that the other three directions of support are determined by this choice. With this we get exactly two cases of type C.

In order to compute the final result, **LER** will do 16 global shifts (called type A [B,C] shifts for cases of type A [B,C]) determining the largest empty rectangle for each case (if it exists) in linear time, respectively. The maximum area empty rectangle as described by theorem 1 is the largest of these 16 (or less) rectangles.
Before we can give the details of type A [B,C] shifts, we need the following definition and lemma.

Definition 2:
Let (x_1,y_1), (x_2,y_2) be two elements of M_i $(i=1,\dots,4)$ with $x_1<x_2$.
(x_1,y_1) and (x_2,y_2) are called "close neighbors of M_i" $:\Leftrightarrow$ there is no other point (x_3,y_3) in M_i with $x_1<x_3<x_2$.

Lemma 3:
Let r be an empty rectangle as described by theorem 1 and $\{t_1,t_2\}=B_i\subseteq M_i$ two supporting points in the same quadrant M_i $(i=1,\dots,4)$ $\Rightarrow$
t_1 and t_2 are close neighbors of M_i.

Proof:
Let w.l.o.g. i=1 and r be an empty rectangle as described by theorem 1 supported by $\{t_1,t_2\}=B_1$ (see fig.5). Assuming there is a point $t'\in M_1$ with x-coordinate between t_1 and t_2, t' will lie inside r, since it is ur-maximal with resp. to S_1' and has distinct y-coordinate - a contradiction. ∎

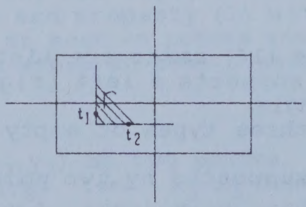


fig.5

With this we can describe the shifts as follows:

Type A shifts
Let's w.l.o.g. take case 3. From lemma 3 we know that all pairs of supporting points of M_3 $[M_1]$ are close neighbors.

Thus, we sort $M_1,\dots,M_4$ by x-coordinate and let each pair of close neighbors of M_3 (represented by the maximum x-coordinate and maximum y-coordinate of both points) shift through M_2 and M_4 to find the rightmost point with smaller y-coordinate and uppermost point with smaller x-coordinate, respectively. With these two points "in mind", we shift each pair of close neighbors of M_3 through M_1 and determine the largest rectangle not containing any point of M_2 or M_4. Pipelining these processes yields linear running time.

Type B shifts

Let's w.l.o.g. take case 2. With the same arguments as given above, we sort $M_1,\dots,M_4$ by x-coordinate. Taking each pair of close neighbors of M_3 as described above, it is easy to see, that the supporting lower point $l_4 \epsilon M_4$ and supporting left point $l_2 \epsilon M_2$ are determined. l_4 is the uppermost point of M_4 with smaller x-coordinate and l_2 is the rightmost point of M_2 with smaller y-coordinate. Thus we let each pair of close neighbors of M_3 shift through M_4 and M_2 and find these both points, respectyvely. With this, a shift through M_1 shows whether this rectangle is empty. Pipelining these processes yields linear running time, too.

Type C shifts

A type C shift is essentially the same, since given the left [lower] supporting point in M_1 the three other supporting points are determined.

3.3. Accumulated Running Time and Space Requirement

Since the the computation of the final result can be done in linear time, the accumulated running time of all divide and merging steps is linear with resp. to n. Each cell has to store a constant amount of information yielding a linear space requirement, too.

REFERENCES

[1] M.J.Atallah, S.E.Hambrusch, SOLVING TREE PROBLEMS ON A MESH-CONNECTED PROCESSOR ARRAY, Report CSD-TR-518, Purdue Univ., West Lafayette, April 1985

[2] B.M.Chazelle, COMPUTATIONAL GEOMETRY ON A SYSTOLIC CHIP, IEEE Trans. on Computers, Vol. C-33, No.9, Sept. 1984

[3] B.Chazelle, R.L.Drysdale, D.T.Lee, COMPUTING THE LARGEST EMPTY RECTANGLE, Proc. Symp. on Theoetical Aspects of Computer Science, 1984, pp 43-54

[4] F.Dehne, $O(n^{1/2})$ ALGORITHMS FOR THE MAXIMAL ELEMENTS AND ECDF SEARCHING PROBLEM ON A MESH-CONNECTED PARALLEL COMPUTER, Techn. Report, Univ. of Wuerzburg, W.-Germany, 1985

[5] A.L.Fisher, H.T.Kung, L.M.Monier, Y.Dohi, A PROGRAMMABLE SYSTOLIC CHIP, Journal of VLSI and Computer Systems, Vol.I, No.2, 1984

[6] H.T.Kung, F.Luccio and F.P.Prparata, ON FINDING THE MAXIMA OF A SET OF VECTORS, J. of the ACM, Vol.22, No.4, Oct. 1975

[7] R.Miller and Q.F.Stout, COMPUTATIONAL GEOMETRY ON A MESH- CONNECTED COMPUTER,Proc.1984 Int. Conf. on Parallel Proc.

[8] A.W.Naamad, W.L.HSU, D.T.Lee, ON THE MAXIMUM EMPTY RECTANGLE PROBLEM, Disc. Applied Math.8, 1984

[9] D.Nassami and S.Sahni, FINDING CONNECTED COMPONENTS AND CONNECTED ONES ON A MESH-CONNECTED PARALLEL COMPUTER, SIAM J. COMPUT., Vol.9, No.4, Nov. 1980

[10] C.D.Thompson and H.T.Kung, SORTING ON A MESH-CONNECTED PARALLEL COMPUTER, Comm. of the ACM, Vol.20,No.4, April 1977

[11] J.D.Ullman, COMPUTATIONAL ASPECTS OF VLSI, Principles of Computer Science Series, Computer Science Press, 1984

POISSON CONFLICT RESOLUTION WITH FEEDBACK BASED ON CONFLICT INTENSITY

NADER MEHRAVARI
AT&T Bell Laboratories
Holmdel, NJ 07733.

ABSTRACT

Communication over a random multiple-access, time-slotted, packet-switched, collision-type, broadcast channel with feedback is considered under a Poisson infinite-user model. The feedback generated by the channel provides some information about the intensity of conflicts. Two conflict resolution algorithms are discussed for the above channel. The proposed algorithms achieve throughputs of 0.334 and 0.369 respectively. The second algorithm is a simple modification of the first. We point out that the modification technique may help to improve the performance of the conflict resolution algorithms operating under other types of feedback.

SUMMARY

In this paper we consider the random multiple-accessing of a time-slotted, packet-switched, collision-type, broadcast channel with feedback under a Poisson infinite-user model. In [1-3] it is assumed that, immediately after each slot, a ternary feedback informs the users whether that slot was empty, contained only one packet, or contained a collision of two or more packets (which must be retransmitted later). This has been referred to as 0,1,e-feedback. Channels with various binary feedbacks have been investigated in [4,5]. The case of d-ary feedback for $d > 3$, where the feedback provides the exact number of collided packets has been investigated in [6,7]. Here we consider a channel feedback which provides the users with partial information about the multiplicity of the conflicts, hence the name "conflict intensity." In particular, assume that associated with user i is a "degree of activeness," d_i, where d_i is a fixed pre-assigned positive real number. In practical situations, this degree of activeness, for example, could represent the energy with which packets are transmitted. The channel feedback, then, for the kth slot is defined as

$$Z(k) = \sum_i d_i \, I_{\{\text{user i transmits in slot k}\}} \tag{1}$$

where $I_{\{A\}}$ is the indicator function of event A. Therefore, $Z(k) = 0$ if none of the users transmit any packets in slot k; and $Z(k) > 0$ if at least one user transmits in slot k. This feedback, for example, could represent the total energy on the channel measured during one slot.

This type of feedback was originally studied in [8] under a Bernoulli finite user model. Depending on whether the value of d_i's are globally known apriori to all users, or whether they have a particular form, the above feedback could lead to interesting situations. For example, if $d_i = 2^i$, then the binary representation of $Z(k)$ provides the exact identity of all active users. However, the discussion and analysis of the proposed algorithms in this paper assumes no prior knowledge of d_i's except that they are positive real numbers and d_i is only known to user i.

In this paper, we have proposed and analyzed two conflict resolution algorithms (CRA) for the above channel. The proposed algorithms achieve throughputs of 0.334 and 0.369 respectively. The first algorithm is a basic interval searching CRA which searches the arrival process for a non-empty interval of length x. After finding such an interval, the CRA enters the conflict resolution state and resolves any conflict among the arrivals in that interval. The second algorithm is a simple modification of the first. It starts by looking for non-empty intervals of length x. After finding two such intervals, the algorithm forms a straddled pair by putting the two non-empty intervals next to each other and then enters the conflict resolution state where it resolves any conflict among the arrivals in the straddled intervals. The higher throughput of the modified CRA is achieved in return for slightly higher delay.

The type of feedback discussed in this paper provides slightly more information than the so-called Something/Nothing (SN) feedback which indicates whether there was any message transmitted in a given slot. Algorithms for a channel with SN feedback and a Poisson infinite user model have been investigated in [5,9]. Note that the throughputs reported in [5,9] can be achieved under the feedback discussed in this paper. The emphasize of this paper is, however, not the magnitude of the throughputs but the method used to improve the throughput of the first algorithm. The hope is to be able to generalize the modification technique for designing non-First-Come First-Served CRA for other types of feedback. (Interested reader is referred to [10] for the full version of this paper.)

REFERENCES

[1] J. I. Capetanakis, "Tree Algorithm for Packet Broadcast Channels," *IEEE Trans. Info. Theory,* Vol. IT-25, No. 5, pp. 505-515, September, 1979.

[2] R. G. Gallager, "Conflict Resolution in Random Access Broadcast Networks," *Proc. of the AFOSR Workshop on Comm. Theory and Applications,* Provincetown, MA, pp. 74-76, September, 1978.

[3] T. Berger, "The Poisson Multiple Access Conflict Resolution Problem," in *Multi-User Communications,* Ed. G. Longo, Springer-Verlag, CISM Courses and Lecture Series, No. 265, 1981.

[4] T. Berger, N. Mehravari, D. Towsley, J. K. Wolf, "Random Multiple Access Communication and Group Testing," *IEEE Trans. on Comm.,* Vol. COMM-32, No. 7, July, 1984, pp. 769-779.

[5] N. Mehravari, T. Berger, "Poisson Random-Multiple Access Contention Problems with Binary Feedback," *IEEE Trans. Infor. Theory,* September, 1984.

[6] B. S. Tsybakov, "Resolution of Conflict with Known Multiplicity," *Problemy Peridachi Informatsii,* Vol. 16, No. 2 pp. 69-82, 1980.

[7] L. Georgiadis and P. Papantoni-Kazakos, "A Collision Resolution Protocol for Random Access Channels with Energy Detectors," *IEEE Trans. Comm.,* Vol. COMM-30, No. 11, pp. 2412-2420, November, 1982.

[8] N. Mehravari, "Random Multiple-Access Comm. with Feedback based on Conflict Intensity," *Proc. 18th Annual Conf. Info. Sci. and Systems,* Princeton Univ., March 1984.

[9] B. Hajek and T. Van Loon, "Decentralized Dynamic Control of a Multi-Access Broadcast Channel," *IEEE Trans. Auto. Control,* June 1982.

[10] N. Mehravari, "Poisson Conflict Resolution Problem with Feedback based on Conflict Intensity," AT&T Bell Laboratories, 1985.

TECHNIQUES FOR ASSESSING THE STABILITY OF MULTIPLE-ACCESS PROTOCOLS

LINN I. SENNOTT
Dept. of Mathematics
Illinois State University
Normal, IL 61761

Many multiple-access protocols can be modeled as Markov chains. Every countable state, irreducible, aperiodic Markov chain falls into one of three classes: a) ergodic (positive recurrent) if the mean return time to each state is finite (in this case the steady-state distribution exists); b) null recurrent if the probability of returning to each state is 1 but the mean return time is infinite; or c) transient if the chain eventually drifts to $+\infty$.

If (X_n) is a Markov chain on $0,1,2,\ldots$ recall that the drift at i, γ_i is given by $\Sigma_j P_{ij}(j-i)$, and Kaplan's Function $\psi_i(z)$ is given by $(z^i - \Sigma_j P_{ij} z^j)/(1-z)$, for $0 < z < 1$. We define for $n \geq 1$

$$\gamma_i^n = \Sigma_j P_{ij}\{j(j-1)\ldots(j-n) - i(i-1)\ldots(i-n)\}.$$

The following results are often useful:

1. If there exist positive integers N and B and a number c $(0 < c < 1)$ such that $\gamma_i \geq 0$ $(i \geq N)$ and $\psi_i(z) \geq -B$ $(i \geq N,\ c \leq z < 1)$, then the chain is not ergodic. [2], [3] (For a multi-dimensional version, see [4]. For a generalization, see [5].)

2. If there exist a positive integer N and a number z_0 $(0 < z_0 < 1)$ such that $\psi_i(z_0) \geq 0$ $(i \geq N)$, then the chain is transient.

3. Fix $n \geq 1$. If there exist positive integers N and B and a number c $(0 < c < 1)$ such that $\gamma_i^n \geq 0$ $(i \geq N)$ and $i^n \psi_i(z) \geq -B$ $(c \leq z < 1)$, then the chain is not ergodic.

The last result may be useful when the drifts are eventually negative but converging to 0, and hence neither Pakes' criterion nor 1. apply.

These results may be used to easily obtain a characterization of the stability of infinite source, controlled slotted ALOHA [1].

THEOREM: (A) If $c_1 \geq c_0$, the chain is transient under every retransmission control policy.

Now assume $c_1 < c_0$, and $\alpha = c_1/c_0 < 1$.

(B) If $\mu > c_0 e^{\alpha-1}$, the chain is transient under every retransmission control policy.

(C) If $\mu < c_0 e^{\alpha-1}$, the chain is ergodic under the optimal retransmission control policy.

(D) If $\mu = c_0 e^{\alpha-1}$, the chain is null recurrent under the optimal retransmission control policy.

In the infinite source exponential back-off protocol, a terminal that has unsuccessfully attempted transmission k times will attempt in the next slot with probability 2^{-k}. If $n = (n_1, n_2, \ldots)$ denotes a state, where n_i is the number of terminals that have unsuccessfully attempted i times, then the conditional throughput δ_n is given by

$$\Pi_i (1-f_i)^{n_i} (c_1 + c_0 \Sigma_i n_i f_i/(1-f_i))$$

and the drift on the backlog is $\mu - \delta_n$.

Letting $g_i = f_i/(1-f_i)$ and noting that $\Pi_i (1-f_i)^{n_i} \le (1/2)^{\Sigma n_i g_i}$, we can upper bound δ_n by $\max_{0 \le x} (1/2)^x (c_1 + c_0 x)$.

In the Poisson case, this implies that the drifts will be non-negative for $\lambda \ge .461$ and hence by the multidimensional analogue of 1. [4], the chain is not ergodic.

REFERENCES

1. G. Fayolle, E. Gelenbe, and J. Labetoulle, "Stability and optimal control of the packet switching broadcast channel," J. ACM, 24, 375-386 (1977).
2. M. Kaplan, "A sufficient condition for nonergodicity of a Markov chain." IEEE Trans. Inform. Th. IT-25, 470-471 (1979).
3. L. Sennott, P. Humblet, and R. Tweedie. "Mean drifts and the non-ergodicity of Markov chains," Oper. Res. 31, 783-789 (1983).
4. L. Sennott, "Tests for the nonergodicity of multidimensional Markov chains," Oper. Res. 33, 161-167 (1985).
5. W. Szpankowski, "Some sufficient conditions for non-ergodicity of Markov chains," J. Appl. Prob. 22, 138-147 (1985).

VARIABLE LENGTH CODING SCHEMES FOR COLLISION-FREE CHANNEL ACCESS

Douglas W. Browning and John B. Thomas
Department of Electrical Engineering
Princeton University

Abstract— **Variable length coding schemes are considered for collision-free access to a single broadcast channel. The schemes are intended to operate efficiently with a large, finite number of users under all traffic conditions, including heavy channel loads with bursty traffic. The method also allows for the establishment of priority users and a short, bounded delay for high priority messages.**

The Context. Collision-free Channel Access refers to the approach of broadcast channel access where users establish the access rights for the channel before transmission begins. Here we will consider a scheme without collision detection where the channel is divided into time slots of some duration α times the length of a packet, with α long enough for any transmission to be detected by all users.

Kleinrock and Scholl [1] considered a distributed polling scheme where, after a packet has completed transmission, each user is assigned to a time slot. If a user is *active* (has a packet to transmit), it waits for the appropriate number of slots, and if no other user has begun transmission, gains access to the channel by beginning to send a packet. The scheme proved to be effective for a small number of users, but less effective when the number of users was large. Rothauser and Wild [2] considered a scheme that would accommodate a large number of users, but there are some operational problems. For example, users must wait until all active users are identified before transmission can begin. Mark [3] considered a scheme where access is resolved on a separate subchannel, and where users access according to an address in a manner similar to the method considered here. With collision detection, the problem of access becomes a tree-polling problem, which has been dealt with by Capetanakis [4] and others.

The Scheme. When a user transmits a packet, all users are assigned a binary code for access to the channel. When the transmission is complete, each active user competes for the channel by transmitting a pulse in the slots corresponding to a 1 in its code and listening to the channel in the slots corresponding to a 0 in its code. If a user detects a pulse in a slot where it has a 0 in its code, it ceases to compete for the channel until the next transmission is complete. If the channel is idle, users begin access by sending a pulse to "activate" the channel. The pulse is then followed by a collision-resolution period as with the busy channel case, except that a fixed code corresponding to the user's address is used. We note in passing that if channel synchronization is not lost during idle periods, it is possible to improve on this slightly by dividing the users into groups and giving only one group the opportunity to access in each slot.

It can be assumed without loss of generality that all decipherable codes used here have all code words ending in 1. The final 1 of each user's address code is replaced by the beginning of the message, saving one time slot in access.

It is possible to implement a high priority, low arrival rate "emergency message" using this scheme by reserving the first slot after a transmission or channel activation for a priority indicator. Following a transmission, users with priority messages access by sending a pulse followed by a code including slots to distinguish between priorities, if there is more than one, and a fixed code corresponding to the user's address, as with idle channel access. A single high priority message will have a small, bounded time to transmission. Implementing such messages is also useful for the correction of erroneous access information during a busy period.

Performance. Figure 1 shows the expected overhead per packet used for access of the channel versus the number of active users in a population of 1000 users. The optimal bounds enclose bounds derived from Huffman coding of the *a priori* distribution of n users distributed uniformly among the population. The curve denoted as a "coded scheme" is the performance of a nonadaptive code (in the sense of not adapting to channel load) derived from Huffman coding and an *a priori* distribution that is a weighted average of distributions with several different numbers of active users. Also shown are results from the MLMA scheme of Rothauser and Wild, and from a simple adaptive CSMA/CD scheme where the probability of transmitting for each user is $(0.5)^k$, where k is the number of collisions since the last packet transmitted, as was first suggested by Metcalfe and Boggs [5].

Figure 2 shows the delay versus throughput for this type of system in comparison with CSMA/CD for α=.1 and α=.01 as well as the lower bound provided by an M/D/1 queue.

Conclusions. When compared with the MLMA scheme of Rothauser and Wild [2], this scheme has the following disadvantages and advantages:

Disadvantages: 1) Users must track which user has access to the channel for the duration of each busy period. 2) Users must determine their new access code after each channel access.

Advantages: 1) The performance is improved, especially at low loads. 2) High priority emergency messages can be implemented with a short, bounded access time.

In relation to CSMA/CD, which is presently the most popular access scheme for broadcast channels, this scheme has the following disadvantages and advantages:

Disadvantages: 1) The performance in terms of delay versus throughput is inferior for all but a high throughput. 2) All users must have an address, and must monitor the channel during a busy period. 3) The channel must have a facility for slot synchronization.

Advantages: 1) The scheme does not require collision detection, and is superior in performance to CSMA without collision detection. 2) All channel loads and conditions can be accommodated. 3) A high priority, low occurrence emergency message can be implemented with a short, bounded access time. 4) The scheme does not require any adaptive adjustments for access, as is necessary for efficient operation of CSMA/CD.

For further details, see [6].

Acknowledgements. This work was supported in part by the U. S. Army under Grant DAAG29-85-K-0191 and in part by the National Science Foundation under Grant ECS83-17777.

References

[1] Leonard Kleinrock and Michel Scholl, "Packet Switching in Radio Channels: New Conflict-Free Multiple Access Schemes for a Small Number of Data Users", *ICC Conf. Rec., Chicago, ILL.*, June 1977, pp. 22.1-105 to 22.1-111.

[2] E. H. Rothauser and D. Wild, "MLMA-A Collision-Free Multi-Access Method", *IFIP77*, B. Gilcrist, ed., pp. 431-436.

[3] Jon Mark, "Distributed Scheduling Conflict-Free Multiple Access for Local Area Communication Networks", *IEEE Trans. Comm.* **28,** p. 1968-1976 (Dec. 1980).

[4] J. I. Capetanakis, "Generalized TDMA: The Multi-Accessing Tree Protocol," *IEEE Trans. Comm.* **27,** p. 1476-1484 (Oct. 1979).

[5] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the ACM,* **19,** No. 7, July 1976.

[6] D. W. Browning, Ph.D. thesis, Princeton University, December 1985, Chapter 2.

Figure 1

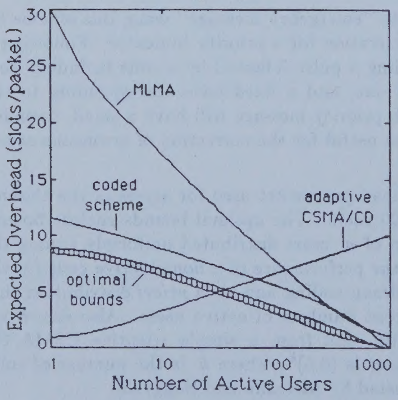
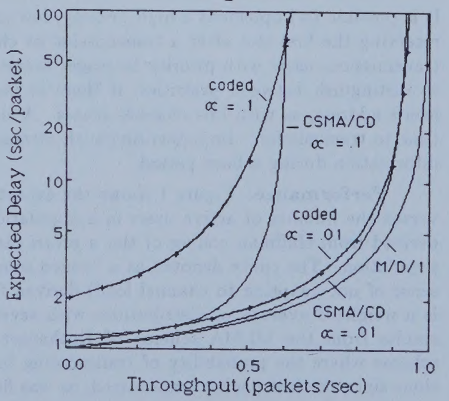


Figure 2



AN ANALYSIS OF THE SILENT CHANNEL ACCESS METHOD

ASHOK P. NADKARNI and KENNETH S. VASTOLA
Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute
Troy, New York 12180-3590 U.S.A.

In this paper, we present a brief summary of our analysis of a new collision-free protocol that implements shared access to a communication channel in a local area network environment. This protocol, called the Silent Channel Access Method (SCAM), belongs to the class of virtual token protocols and allows the users to access a shared channel in a collision free manner. SCAM is described in more detail in [2,3]. Where it differs from previous virtual token schemes (see the survey by Fine and Tobagi [1]) is that the algorithm followed by each node is simple, depends solely on the state of the channel as observed by the node, does not require explicit synchronization between nodes and is very generally applicable. In fact, the only assumptions we make about the environment in which the protocol is to be used are:

1) The network is of a broadcast type (e.g., ring, bus, packet radio) in which there is one and only one transmission path between every two nodes.

2) The propagation delays satisfy the triangle inequality; that is, for any three nodes i, j and k, we have

$$\tau_{ij} + \tau_{jk} \geq \tau_{ik}$$

where τ_{ij} represents the propagation delay from node i to node j. The idea underlying the SCAM protocol is basically the same as for other virtual token protocols with silence being used as a token to indicate the transfer of transmission rights. A number of control strategies that accomplish this have been proposed before but they all suffer from at least one significant drawback as discussed in [2,3]. The SCAM protocol, on the other hand, uses a channel access mechanism that is efficient and simple as well as more generally applicable than any other virtual token protocol. This protocol mechanism is based on the fact that *in the absence of collisions, the proportion of time that the channel is heard to be idle is exactly the same for each node independent of its position and independent of who transmits.* This concept is developed in [2,3] and a proof is given in [2]. The protocol could be implemented using timers at each node which run only when the channel is heard to be idle. When the appropriate amount of idle time has elapsed a packet is sent if desired and the timer is reset. For startup and a discussion of other issues, see [2,3].

ANALYSIS

Assumptions (1) and (2) above are the only ones necessary for implementing SCAM. However, we make a few additional assumptions only for the purpose of simplifying our analysis.

3) The arrival process to each of the N nodes on the network is Poisson. Furthermore, the arrival process at each node is independent of those at all other nodes.

4) The propagation delays between users adjacent in the transmission order are all equal. In other words, $\tau_{i,i+1} = \tau$ for all i.

Under these assumptions, we can derive the throughput-delay relationship of the SCAM protocol [2]. We first calculated this for the case where packets are of fixed length and require a constant time T for transmission. We then repeated this for packet lengths drawn from a bimodal distribution in order to illustrate how this method might be used for any discrete packet-length distribution (see [2]).

In Fig. 1, the throughput-delay equation has been plotted for various values of τ with the number of stations N assumed to be 100. At $\tau = .001$, the performance is seen to be extremely good. For example, for throughput rates up to .9, the average delay remains below twice the packet transmission time T and even with a throughput of .99, the average delay is just 7T.

We next looked at the effect of the number of nodes on the performance of the SCAM protocol. Not surprisingly, we found that as N increases the network performance goes down if the other parameters are held fixed. This corresponds to increasing the overall size of the network (e.g., a 1 km cable with 100 nodes becomes a 2 km cable with 200 nodes). On the other hand, if we want the same network with more nodes (e.g., 1 km cable with 200 nodes), we have to decrease the internode propagation delay τ proportionately (divide τ by 2 in our example). Thus keeping the product $N\tau$ constant while increasing N should result in little change in performance (unlike, for example CSMA). Fig. 2 shows this to be the case.

It can be shown [2] that given a network which could use SCAM to achieve the best performance curve in Fig. 1 (i.e., N = 100, T = 1, $\tau = .001$), that same network using BRAM or other early token protocols (see [1]) would only achieve (approximately) the worst performance curve in Fig. 1 ($\tau = .05$). This is because the time between consecutive turns in these protocols is the end to end propagation delay which should be approximately $N\tau/2$ (or 50τ in this example). Other more recent virtual token protocols such as L-express-net, Silentnet and BID have performance similar to SCAM; however, SCAM has distinct implementation advantages over these such as limited storage requirements, topological generality, and completely distributed operation (see [2,3]).

1. M. Fine and F.A. Tobagi, *IEEE Trans. Computers*, Vol. C-33, Dec. 1984.
2. A.P. Nadkarni, Master's Thesis, ECSE Dept., RPI, Troy, NY, 1985.
3. A.P. Nadkarni and K.S. Vastola, *IEEE GLOBECOM '85*, New Orleans, LA, Dec. 1985.

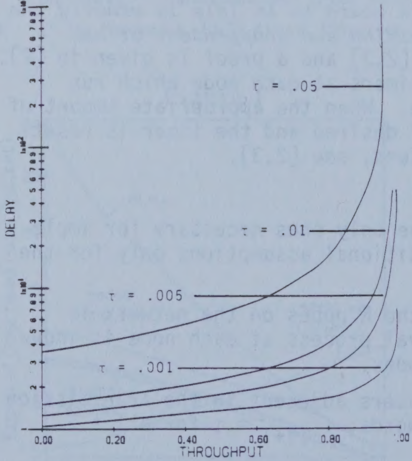


Fig. 1. SCAM Throughput-Delay Characteristic: N = 100 (T = 1).

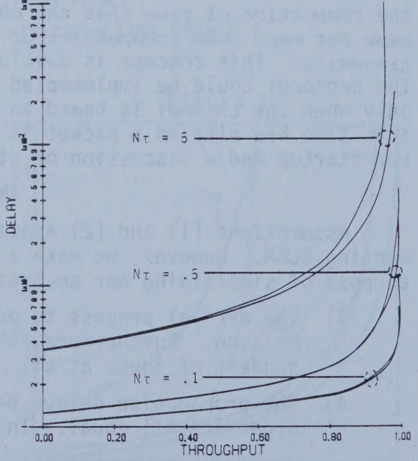


Fig. 2. SCAM Throughput-Delay Characteristic: Constant $N\tau$ (T=1).

AN ANALYSIS OF A CSMA/CD COLLISION RESOLUTION SCHEME

YIH-CHIAO LIU
Department of Electrical and Computer Engineering
University of Texas at Austin
Austin, Texas 78712

and

GARY L. WISE
Departments of Electrical and Computer Engineering and Mathematics
University of Texas at Austin
Austin, Texas 78712

ABSTRACT

In this paper we analyze the performance characteristics of a packet broadcasting random multiple access computer communication network with a CSMA/CD protocol. The analysis is based on the Enet II protocol, which was designed to effectively resolve collisions in such a network. We establish bounds on the performance of the network.

I. INTRODUCTION

In the past few years, packet broadcasting random multiple access computer communication networks have become commercially available. A typical example of such a network is the Ethernet developed by Xerox, which was designed based on the idea of carrier sense multiple access with collision detection (CSMA/CD) [1]. The basic Ethernet protocol is described in the IEEE standard 802.3, where a station among a number of users sharing a common broadcast channel will listen before transmitting, and defer if the channel is busy. In Ethernet, when two or more stations collide, a retransmission is scheduled for a later time. Each colliding station waits for a random period of time before retransmitting. The mean of this randomized waiting period before retransmission is increased under times of heavy load in order to sustain channel efficiency. Although Ethernet has the advantage of easy interconnection of stations to the common channel and it provides a high level of utilization of the channel [2], it does not truely address the problem of how to effectively resolve collisions in the channel. Thus packets involved in a collision may incur excessive delay due to waiting and abortion of transmission. This is even less desirable if a large number of stations are connected to the network and channel access demand is high.

II. PRELIMINARIES

Recently, a protocol called Enet II was introduced by Molloy [3] as a candidate for the second generation of Ethernet. This protocol is designed to effectively resolve contention in a broadcast multiple access network such as Ethernet. Before we describe the Enet II protocol, we will introduce some notation. We will assume that the "diameter" of the network is given, that is, the maximum propagation delay between any two stations in the network. Let r be twice the diameter of the network. Any station listening to the channel for an interval of r units of time after transmitting would be guaranteed to hear something if anyone else were attempting to use the channel during that interval of time. A collision occurs when two or more stations attempt to transmit within an interval of $r/2$ units of time. According to the protocol, the stations are in one of three states: inactive, active, and deferred. Inactive stations either do not have anything to send or have just finished sending something. Active stations are trying to send a packet (which might be a new message or might be a message involved in a previous collision). Deferred stations have attempted to transmit but are waiting for the active stations to leave the active state. We also assume that each station has a coin flipping mechanism such that the probability of head appearing is p. The Enet II protocol is given by the following procedure.

```
Inactive stations (with a packet generated):
    Follow normal CSMA procedure (i.e., check channel before going to active state).
    If channel is idle, wait for 3r units of time; then transmit.
    If channel is busy, wait until it is idle for 3r units of time; then transmit, and the station goes to the active state.
Active stations:
    If transmission is successful, station returns to the inactive state.
    If a collision occurs, all participants in that collision flip a coin with probability of head equalling p.
        If "head" appears, the station tries to transmit again.
        If "tail" appears, the station monitors the channel passively:
            if the station sees the channel idle for r units of time, transmit;
            if the station sees a successful transmission, wait for the end of that transmission and then transmit; and
            if the station sees a collision, the station changes to the deferred state.
Deferred stations:
    Passively monitor the channel.
    If the station sees the channel idle for an interval of 2r units of time, it transmits and returns to the active state.
    If the station sees the channel as not idle in an interval of 2r units of time, it remains in the deferred state.
```

The Enet II protocol is simple and needs no extensive support, such as clocks, addresses, current load estimates, or preassigned orderings, as compared with some other contention resolution protocols [4,5]. This protocol is characterized by the introduction of a "gate" for new arrivals such that stations have to wait for the channel to be idle for a period of 3r units of time before transmitting a new packet. Therefore, stations need not monitor the channel when they have nothing to send. All new arrivals must stay behind that gate until all active or deferred users, if any, are finished. Similarly, the deferred users must stay behind their gate for a period of 2r units of time until the active users are finished. Assuming at least one success and at least one failure among the Bernoulli trials generated by the active users, the random test mechanism will decrease the number of active users participating in a collision by successfully transmitting some or having them move to a deferred state in the case that it is known that two or more stations are still in the active state. Active stations which flipped "tail" in their coin flipping tests still transmit after the channel is free for r units of time, effectively announcing their presence to keep deferred stations from erroneously concluding that all active stations are done. When all of the active stations transmit successfully, all of the deferred stations will change to the active state.

III. DEVELOPMENT

In [6], we performed a preliminary analysis of the Enet II protocol by assuming a simplified model of n stations in the network, where each station generates packets of equal length. In this section, we will investigate the performance of the Enet II protocol while stations in the network generate packets of variable packet lengths.

We assume that there are n stations in the network, and we index these stations from 1 to n. Assume that each of these n stations is equally likely to have a packet ready to send out and that the packet arrivals at the different stations are mutually independent. In the context of no collision, let τ_i denote the time from when the i-th station transmits a packet until the time that packet exits the channel. When the i-th station involved in a collision successfully transmits its packet, let τ_i denote the time from when the packet is last transmitted by the i-th station until the time that packet exits the channel. We will call the τ_i's the packet transmission times, and we will model τ_i, $1 \leq i \leq n$ as a random variable. Note that this does not exclude the possibility of them being constants. Assume that the τ_i's are mutually independent and that the minimum of the support of each random variable τ_i is always greater than r/4. The latter assumption guarantees that, in case of a collision, the time needed to witness the collision is less than the time needed to resolve the collision. (In local area networks, the usual maximum spatial separation among the stations is less than 10 Km, and the assumption that $\tau_i > r/4$ is usually easily satisfied.)

A station having a packet ready to send listens before transmitting. As we have noted, if more than two stations try to transmit their messages within a period of r/2 units of time, a collision is said to have occurred. A collision will be detected and transmission of all colliding stations will be aborted. Then the stations will flip coins and retransmit according to the outcome of the coins. Let the random variable Z denote the time between when the first packet was sent in a collision and the time when the colliding stations acknowledge the collision and flip coins. We assume that the mean of Z is independent of the number of stations involved in a collision. Let $\delta = E[Z]$.

In the context of the resolution of a collision, assume that there are j ($j \geq 1$) stations monitoring a transmission and waiting for it to end and for the channel to be free. We will let μ_j be the average time until the first of these j waiting stations witnesses the end of the current transmission. In this context we use the word average in the sense that the average is over all possible combinations of j waiting stations and one transmitting station. For each combination there is an associated time for the first to witness the end of the current successful transmission. There is also a probability associated with each combination. This probability is obviously dependent upon the packet arrival statistics, and it is also dependent on the coin flipping outcomes. We note that $\mu_j \leq r/2$, $1 \leq j \leq n$. We will let $\mu_0 = 0$.

Consider the successful transmission of a packet. It is transmitted either without having experienced a collision or during the resolution of a collision. In the absence of a collision, the average time required for a successful transmission will be denoted by C_1, and is given by

$$C_1 = \frac{1}{n} \sum_{i=1}^{n} E[\tau_i],$$

since each of the stations was assumed to be equally likely to have a packet ready to send out. If the packet is transmitted during the resolution of a collision, then the transmission occurred when the transmitting station flipped "head" and each of the others either flipped "tail" or was in the deferred state. In a collision, each station is as equally likely to be involved, and due to the coin flipping mechanism each station in the collision is as equally likely to be the one transmitting during a successful transmission. Thus C_1 given above is the average time to execute a successful transmission either during a collision or in the absence of one.

In [6], it is assumed that all packets transmitted in the channel are of the same length, and it takes τ units of time to transmit a packet without experiencing a collision. The results in [6] are therefore a special case of the following analysis by taking $\tau_1 = \ldots = \tau_n = \tau$, and hence $C_1 = \tau$.

Consider a k-way collision, $2 \leq k \leq n$, and let C_k be the average time between when the first packet was sent but ends up in a k-way collision and when the very last packet in this collision is successfully transmitted. The average is taken over all possible choices of k colliding stations of n stations in the network. Then by using the law of total probability, C_k is given by

$$C_k = C(k,0,0) \quad \text{for } k \geq 2,$$

where the C(i,j,k)'s satisfy the following difference equations

$$C(0,0,0) = 0$$

$$C(0,j,k) = r + C(j,0,k), \qquad \text{for } j \geq 1, \tag{1}$$

$$C(0,0,k) = \mu_k + 2r + C(k,0,0), \quad \text{for } k \geq 1, \tag{2}$$

$$C(1,j,k) = C_1 + \mu_j + C(j,0,k), \tag{3}$$

where, for $i \geq 2$,

$$C(i,0,k) = \left[\delta + \sum_{\ell=1}^{i-1} \binom{i}{\ell} p^{\ell}(1-p)^{i-\ell} C(\ell,i-\ell,k) + r(1-p)^i\right] / \left[1-p^i-(1-p)^i\right], \tag{4}$$

and

$$C(i,j,k) = \delta + \sum_{\ell=1}^{i-1} \binom{i}{\ell} p^{\ell}(1-p)^{i-\ell} C(\ell,i-\ell,j+k) + \left[p^i+(1-p)^i\right]C(i,0,j+k) + r(1-p)^i. \tag{5}$$

The arguments of C(i,j,k) can be interpreted as i corresponding to the number of active stations ready to transmit their messages, j corresponding to the number of active stations flipping "tail" after a collision and passively monitoring the channel, and k corresponding to the number of stations in the deferred state; and C(i,j,k) can be interpreted as the average time between when the situation corresponding to the arguments first occurs in executing the Enet II protocol until the last packet in this situation is successfully transmitted. In the context of (5), two or more stations attempt to transmit (i.e. $i \geq 2$); it takes an average of δ units of time to detect the collision, and the remaining terms in (5) represent outcomes of coins flipped by the contending i stations and the corresponding average times to resolve the situations associated with these outcomes. Equation (4) is a special case of (5) obtained by setting the second argument to zero. The boundary conditions given by (1)-(3) are obtained according to the protocol. Equation (1) represents the case where j stations flipping "tail" see the channel being free for r units of time and try to gain access to the channel; the k deferred stations remain in the deferred state in this case. Equation (2) represents the case where upon sensing the completion of a successful transmission and the ensuing idleness of the channel, the k stations wait 2r units of time and then attempt to access the channel; recall that μ_k is the average time until the first of k stations witnesses the completion of the successful transmission. Equation (3) corresponds to the case where one station is transmitting; it takes an average of C_1 units of time to finish this single transmission and then it takes an average of μ_j units of time until the first of the j stations flipping "tail" witnesses the end of the successful transmission; these j stations attempt to access the channel and the k deferred stations remain in the deferred state.

From (1)-(5) we can obtain C_k, $2 \leq k \leq n$. For example, $C_2 = \mu_1 + 2C_1 + [\delta+r(1-p)^2]/[2p(1-p)]$. For $k \geq 3$, C_k can be calculated recursively by aid of a computer. From the recursiveness of (1)-(5), we observe that C_k, $2 \leq k \leq n$, is a positive function of p in the open interval (0,1). Also, C_k is $+\infty$ if p is equal to 0 or 1, since in either case a k-way collision can never be resolved. By a limiting argument, we can see that C_k approaches $+\infty$ as p approaches 0 and 1. Thus C_k has a minimum and can be minimized by choice of p. Note also that C_k is always bounded by kC_1, which is the average time to transmit k packets sequentially and successfully. We will call C_k-kC_1 the average collision resolution time since this is the average of the extra time not accounted for in the actual transmission but rather in resolving the collision. Since C_2-2C_1 is not dependent upon C_1, it follows from the recursive nature and an induction argument that the average collision resolution time C_k-kC_1, $2 \leq k \leq n$, is independent of C_1. Hence C_k is a sum of two terms: the average overall time to transmit k packets sequentially, and the average collision resolution time. In Fig. 1 we present a plot of the average collision resolution time for various values of k. In this plot, we assume that C_1, the average time to transmit a randomly chosen packet, is fixed. Note that this plot for average collision resolution time is independent of C_1. We also observe that the minimizing p for each C_k-kC_1 is different and is not equal to 1/2 for any k.

Another factor on which C_k depends is δ, the average time from when the first colliding packet is sent until the coins are flipped. Obviously δ depends upon the characteristics of the individual facilities in practice. In Fig. 2, we present a plot of C_4-4C_1 versus p for various values of δ. We observe that for larger δ, the average collision resolution time C_4-4C_1 is longer. We also note that it follows from the preceding recursive equations that for a fixed p, C_k is an affine function of δ for $k \geq 2$.

We note that in (2) and (3), μ_j, $1 \leq j \leq n$, is upper bounded by r/2. Due to the recursive nature of (1)-(5), we can upper bound C_k, $2 \leq k \leq n$, by C_k^*, which is obtained through (1)-(5) by setting μ_j and μ_k each equal to r/2 in (2) and (3), respectively.

In the context of a k-way collision ($2 \leq k \leq n$), we will now consider the situation where a particular station involved in the collision, say station m, is concerned with how long it will take to successfully transmit its packet. Let L_k be the average time from when the first packet involved in a k-way collision was transmitted until when the packet of interest is successfully sent. This average is again taken over all possible choices of k colliding stations including the station of interest among all n stations in the network. Then by the use of the law of total probability and a similar argument used in obtaining the C_k's, L_k, $2 \leq k \leq n$ is given by

$L_k = L(\underline{k},0,0)$ where $L(\underline{k},0,0)$ satisfies the following recursive equations:

$$
\begin{aligned}
L(0,0,0) &= 0,\\
L(0,0,\underline{k}) &= \mu_k + 2r + L(\underline{k},0,0), \quad \text{for } k \geq 1,\\
L(\underline{1},j,k) &= E[\tau_m]\\
L(1,\underline{j},k) &= C_1 + \mu_j + L(\underline{j},0,k), \quad \text{for } j \geq 1,\\
L(1,j,\underline{k}) &= C_1 + \mu_j + L(j,0,\underline{k}), \quad \text{for } k \geq 1,
\end{aligned}
$$

where for $i \geq 2$,

$$
\begin{aligned}
L(\underline{i},j,k) &= \delta + L(\underline{i},0,j+k),\\
L(i,\underline{j},k) &= \delta + L(i,0,\underline{j+k}),\\
L(\underline{i},0,k) &= \Big\{ \delta + \sum_{\ell=1}^{i-1} p^{\ell}(1-p)^{i-\ell} \Big[\binom{i-1}{\ell-1} L(\underline{\ell},i-\ell,k)\\
&\qquad + \binom{i-1}{\ell} L(\ell,\underline{i-\ell},k)\Big] + r(1-p)^i \Big\} / [1-p^i-(1-p)^i],\\
L(i,0,\underline{k}) &= \Big\{ \delta + \sum_{\ell=1}^{i-1} \binom{i}{\ell} p^{\ell}(1-p)^{i-\ell} L(\ell,0,\underline{i-\ell+k}) \Big\} / [1-p^i-(1-p)^i].
\end{aligned}
$$

In the above equations, we use an underline to represent where the station m with the packet of interest lies among the three classes of stations consisting of those who are competing for transmission, those who had flipped "tail" and are passively monitoring the channel, and those who are in the deferred state. Similar observations and arguments in obtaining the C_k's can be applied to the recursive equations of $L_k = L(\underline{k},0,0)$. One can show that L_k is also a continuous function of p in the open interval (0,1), and $L_k = +\infty$ when p is either 0 or 1. Also, L_k approaches $+\infty$ as p approaches 0 or 1. Hence a minimum of L_k, $2 \leq k \leq n$, exists. Similar to the fact that $C_k - kC_1$ is independent of C_1, $L_k - \{E[\tau_m] + (k-1)C_1/2\}$ is independent of either $E[\tau_m]$ or C_1. Consider the case where k packets including the packet of interest are transmitted sequentially; then the average time until the packet of interest is transmitted is given by $E[\tau_m] + (k-1)C_1/2$. Thus L_k is always lower bounded by $E[\tau_m] + (k-1)C_1/2$. We will call the term $L_k - \{E[\tau_m] + (k-1)C_1/2\}$ the average collision delay time for station m. In Fig. 3 we present a plot for the average collision delay time for station m for various values of k where we assume that $E[\tau_m]$ and C_1 are fixed.

We will now investigate the efficiency of the Enet II protocol. As in [1] we will define the efficiency to be the ratio of the average time to transmit a packet without having experienced a collision to the average time to successfully transmit a packet in general. Let P_1 be the probability of a packet being transmitted and experiencing no collision. Let P_k be the probability that a packet being transmitted experiences a k-way collision. Then the efficiency $\mathcal{E}$ is given by

$$\mathcal{E} = \frac{C_1}{C_1 P_1 + \sum_{k=2}^{n} P_k \, C_k/k} \tag{6}$$

Notice that we can always lower bound $\mathcal{E}$ by upper bounding the denominator.

Consider the denominator of (6). It is obvious that maximum efficiency is achieved by minimizing the denominator. Note that in the expression of the denominator, the P_k's depend on the packet arrival statistics, and in practice, determination of the P_k's is often seen as a challenging problem. (It is noted [2] that even if the packet arrivals were assumed to be independent and follow a Poisson arrival distribution, a CSMA/CD protocol can exhibit a packet transmission pattern with little similarity to a Poisson process.) We are now going to find an upper bound for the denominator of (6) by the use of some characteristics of the P_k's and C_k's. Note that the P_k's take values between 0 and 1 regardless of the packet arrival distribution. Hence we have the following inequality for the minimum of the denominator

$$C_1 \le \min_p \left\{ \sum_{k=1}^{n} P_k C_k/k \right\} \le \max_k \left\{ \min_p C_k/k \right\}.$$

Let

$$\mathcal{E}_B = \frac{C_1}{\max_k \left\{ \min_p C_k/k \right\}} ;$$

then the maximum efficiency $\mathcal{E}_{max}$ satisfies the following inequality:

$$\mathcal{E}_B \le \mathcal{E}_{max} \le 1.$$

In Fig. 4, we present plot of C_k/k as a function of p for various values of k. We observe in this case that $\min_p C_k/k$ increases as k increases, and as k gets large, $\min_p C_k/k$ tends to be close to $\min_p C_{k+1}/(k+1)$. We can determine $\mathcal{E}_B$ numerically by consulting Fig. 4 for this case; that is, if $n=8$, $C_1=20$, $\delta=2$, and $r=1$, then $\mathcal{E}_B$ is greater than 83%. However, if we know the prior probability that a packet is transmitted without experiencing a collision, we can further lower bound the maximum efficiency. That is, if P_1 is fixed, then

$$\min_p \left\{ \sum_{k=1}^{n} P_k C_k/k \right\} = P_1 C_1 + \min_p \left\{ \sum_{k=2}^{n} P_k C_k/k \right\} \le P_1 C_1 + (1-P_1) \max_k \left\{ \min_p C_k/k \right\}.$$

In the situation represented by Fig. 4, if we assume that $P_1 = 0.5$, then we have that $0.908 < \mathcal{E}_{max} \le 1$. Alternatively, if we know a lower bound on P_1 we can repeat a similar argument as above.

Recall that $C_1^* = C_1$ and that for $k \ge 2$, $C_k^* \ge C_k$, where the C_k^*'s are obtained by upper bounding the μ_j's by $r/2$. Let P^* be a lower bound for P_1. Then based on the above, we have that $\mathcal{E}_B^* \le \mathcal{E}_{max} \le 1$, where

$$\mathcal{E}_B^* = \frac{C_1}{C_1 P^* + (1-P^*) \max_k \left\{ \min_p C_k^*/k \right\}} .$$

Notice that as the average packet transmission time C_1 increases, the lower bound $\mathcal{E}_B^*$ on the maximum efficiency $\mathcal{E}_{max}$ increases. It would be reasonable to anticipate that with longer packets, the time lost to collision resolution becomes small compared to the packet transmission time, and thus the efficiency would increase. For example, with $P^*=0$, $\delta=2$, $r=1$, and with $n=10$, we have that $\mathcal{E}_B^* > 68.7\%$ for $C_1=10$, $\mathcal{E}_B^* > 81.4\%$ for $C_1=20$, $\mathcal{E}_B^* > 89.7\%$

for $C_1=40$, and $\mathcal{E}_B^* > 94.6\%$ for $C_1=80$; for n=20, we have $\mathcal{E}_B^* > 66.1\%$ for $C_1=10$, $\mathcal{E}_B^* > 79.6\%$ for $C_1=20$, $\mathcal{E}_B^* > 88.6\%$ for $C_1=40$, and $\mathcal{E}_B^* > 93.9\%$ for $C_1=80$. For $P^*=0.5$, $\delta=2$, r=1, and with n=10, we have that $\mathcal{E}_B^* > 81.4\%$ for $C_1=10$, $\mathcal{E}_B^* > 89.7\%$ for $C_1=20$, $\mathcal{E}_B^* > 94.6\%$ for $C_1=40$, and $\mathcal{E}_B^* > 97.2\%$ for $C_1=80$; for n=20, we have $\mathcal{E}_B^* > 79.6\%$ for $C_1=10$, $\mathcal{E}_B^* > 88.6\%$ for $C_1=20$, $\mathcal{E}_B^* > 93.9\%$ for $C_1=40$, and $\mathcal{E}_B^* > 96.9\%$ for $C_1=80$. Finally, for $P^*=0.9$, $\delta=2$, r=1, and with n=10, we have $\mathcal{E}_B^* > 95.6\%$ for $C_1=10$, $\mathcal{E}_B^* > 97.7\%$ for $C_1=20$, $\mathcal{E}_B^* > 98.8\%$ for $C_1=40$, and $\mathcal{E}_B^* > 99.4\%$ for $C_1=80$; for n=20, we have $\mathcal{E}_B^* > 95.1\%$ for $C_1=10$, $\mathcal{E}_B^* > 97.5\%$ for $C_1=20$, $\mathcal{E}_B^* > 98.7\%$ for $C_1=40$, and $\mathcal{E}_B^* > 99.3\%$ for $C_1=80$. It might be interesting to compare our results with the results in [2] where a "typical" Ethernet performance is measured and presented.

IV. CONCLUSION

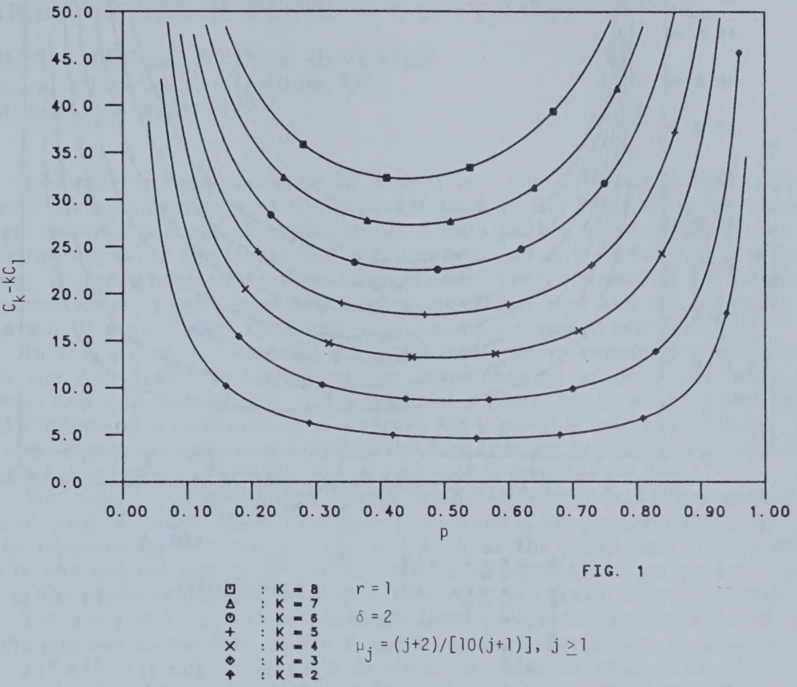
In this paper, we presented an analysis of a CSMA/CD collision resolution scheme, namely, the Enet II protocol. We gave recursive expressions for the average collision resolution time and for the average collision delay time of a collision involving k stations which transmit packets of various packet lengths. We also presented calculations of the lower bounds of the maximum efficiency of the Enet II protocol. We would like to point out that the model we studied is an asynchronous network; and, although we assumed the independence of the packet arrivals at the stations, the analysis is nonparametric in the sense that the result is obtained without assuming any specific packet arrival distribution.

ACKNOWLEDGEMENT

This research was supported by the Air Force Office of Scientific Research under Grants AFOSR-81-0047 and its successor.

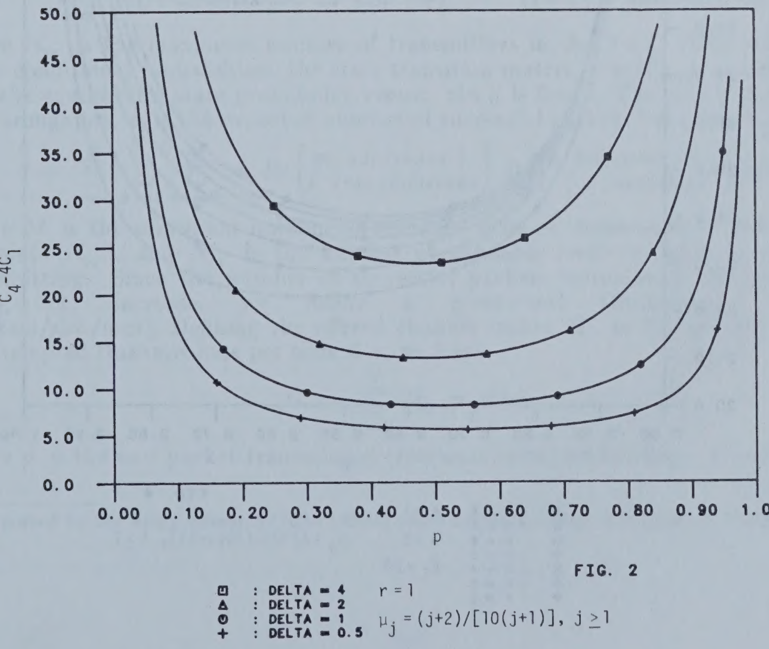
REFERENCES

1. R.M. Metcalfe and D.R. Boggs, "Ethernet: distributed packet switching for local computer networks," *Commun. Ass. Comput. Mach.*, vol. 19, pp. 395-403, 1976.
2. J.F. Shoch and J.A. Hupp, "Measured performance of an Ethernet network," *Commun. Ass. Comput. Mach.*, vol. 23, pp. 711-721, 1980.
3. M.K. Molloy, "Collision resolution on the CSMA/CD bus," *Proc. 9th Conf. on Local Computer Networks*, Minneapolis, MN, Oct. 8-10, 1984, pp. 44-47.
4. R. Gallager, "Conflict resolution in random access broadcast networks," *Proc. AFOSR Workshop in Commun. Theory and Applic.*, Sept. 1978, pp. 74-76.
5. Capetanakis, "Tree algorithms for packet broadcast channels," *IEEE Trans. Inform. Theory*, vol. IT-25, pp. 505-515, Sept. 1979.
6. Y.-C. Liu and G.L. Wise, "A performance analysis of a CSMA/CD protocol," *Proc. 28th Midwest Symp. on Circuits and Systems*, Louisville, KY, Aug. 19-20, 1985.



50.0
45.0
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
$C_k - kC_1$
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
p
⊡ : K = 8
△ : K = 7
⊙ : K = 6
+ : K = 5
× : K = 4
◈ : K = 3
✦ : K = 2
r = 1
$\delta = 2$
$\mu_j = (j+2)/[10(j+1)],\ j \geq 1$

FIG. 1



50.0
45.0
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
$C_4 - 4C_1$
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
p
⊡ : DELTA = 4
△ : DELTA = 2
⊙ : DELTA = 1
+ : DELTA = 0.5
r = 1
$\mu_j = (j+2)/[10(j+1)],\ j \geq 1$

FIG. 2

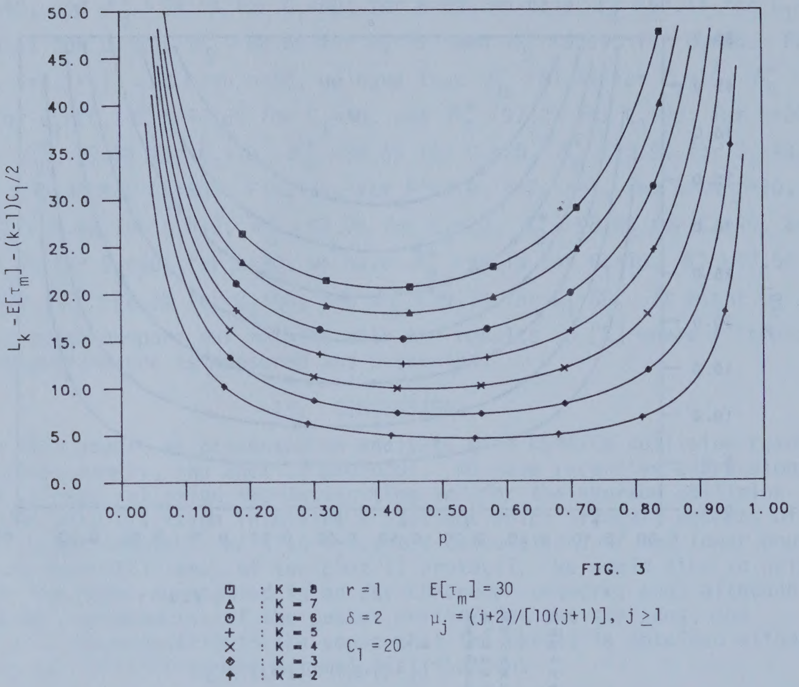
50.0
45.0
40.0
35.0
30.0
25.0
20.0
15.0
10.0
5.0
0.0
$L_k - E[\tau_m] - (k-1)C_1/2$
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
p
▣ : K = 8
△ : K = 7
● : K = 6
+ : K = 5
× : K = 4
◆ : K = 3
✦ : K = 2
$r = 1$
$\delta = 2$
$C_1 = 20$
$E[\tau_m] = 30$
$\mu_j = (j+2)/[10(j+1)],\ j \geq 1$

FIG. 3

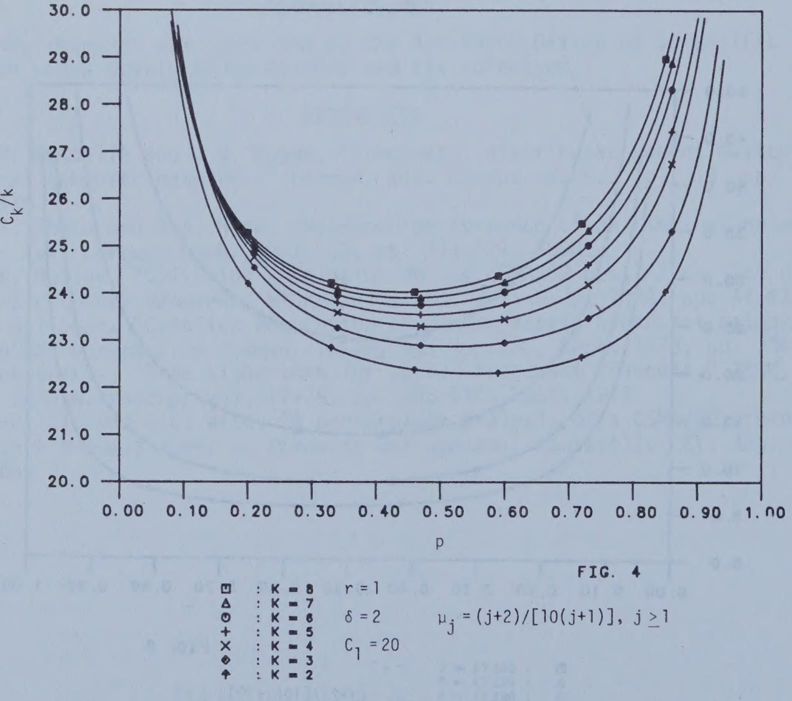
30.0
29.0
28.0
27.0
26.0
25.0
24.0
23.0
22.0
21.0
20.0
C_k/k
0.00
0.10
0.20
0.30
0.40
0.50
0.60
0.70
0.80
0.90
1.00
p
▣ : K = 8
△ : K = 7
● : K = 6
+ : K = 5
× : K = 4
◆ : K = 3
✦ : K = 2
$r = 1$
$\delta = 2$
$C_1 = 20$
$\mu_j = (j+2)/[10(j+1)],\ j \geq 1$

FIG. 4

THE EFFECT OF ACKNOWLEDGEMENTS ON THE PERFORMANCE OF DISTRIBUTED SPREAD SPECTRUM PACKET RADIO NETWORK*

SAMUEL S. LEE and JOHN A. SILVESTER
Communications Sciences Institute, USC
Los Angeles, CA 90089-0272

For reliable communication of data it is common to require the receiver to transmit an acknowledgement (Ack) packet back to the transmitter for those data packets received without checksum error. A data packet transmission is considered successful only when the transmitter receives a correct Ack packet within a time-out interval. If transmitted packets are received with checksum error at the receiver due to channel noise or multi-user interference, no Ack is sent and the transmitter will retransmit its packet after a random delay to avoid repeated collisions [2,4].

In this paper, we consider a slotted multi-access communication system in which the Ack traffic is carried on the same channel as data and analyze two schemes: non split-slot scheme; and a split-slot scheme. In the non split-slot scheme, the slot following a transmission is reserved for a possible acknowledgement. In the split-slot scheme, advantage is taken of the fact that an Ack is much shorter than a data packet, so only the first portion of each slot is reserved for Acks.

We consider a distributed Slotted-Aloha [1] spread spectrum packet radio network with N users, where each user is assigned a unique spreading code. Nodes can be transmitters or receivers, but not both at the same time. The number of Acks in the current slot indicates the number of successful data packet transmissions in the previous slot and we assume that Acks are always received successfully.

Let $x(t)$ and $x(t+1)$ be random variables representing the number of data packets received successfully in slot t and $t+1$, respectively. The sequence of states $x(t)$, $x(t+1)$, $x(t+2)$, ... forms a discrete-time Markov chain with the one-step transition probability, $P_{nm} = Pr\,[x(t+1) = m \mid x(t) = n]$, which is given by:

$$P_{nm} = \sum_{k=m}^{N_{tx}} Pr\left[\begin{matrix} m \ \textit{successes in slot} \ t+1 \mid k \\ \textit{transmissions in slot} \ t+1 \end{matrix}\right] Pr\left[\begin{matrix} k \ \textit{transmissions in slot} \\ t+1 \mid n \ \textit{successes in slot} \ t \end{matrix}\right]$$

where N_{tx} is the maximum number of transmitters in slot $t+1$. After solving for these conditional probabilities, the state transition matrix, $P = [P_{nm}]$, is constructed and the equilibrium state probability vector, $\pi(n)$, is found. The steady-state channel throughput, β, is the expected number of successful packets per slot, i.e.,

$$\beta = \sum_{n=0}^{N} \sum_{k=1}^{N_{tx}} \sum_{m=0}^{M} m \ Pr\left[\begin{matrix} m \ \textit{successes} \mid \\ k \ \textit{transmissions} \end{matrix}\right] Pr\left[\begin{matrix} k \ \textit{transmissions} \\ \mid n \ \textit{successes} \end{matrix}\right] \pi(n)$$

where M is the maximum number of successes given k transmissions and is given by $\min(k, N_{rx})$, and N_{rx} is the number of available receivers when k users are transmitting. Since the number of successful packets increases as the number of users, N, increase, we define a normalized throughput, $S = \beta / N$ (packets/slot/user). Defining the offered channel traffic, G, as the average number of attempted transmissions per time slot, we have

$$G = \sum_{n=0}^{N} N_{tx} \ p \ \pi(n)$$

where p is the new packet transmission (retransmission) probability. The delay, D, in terms of the average number of retransmissions per successful packet is given by $D = (G - \beta) / \beta$.

---

* Supported by the Army Research Office (ARO) under contract number DAAG29-84-K-0084.

Several different Ack schemes can be considered.

Non Split-Slot Scheme. In this scheme, users who received a packet during the current slot without checksum error transmit an Ack using the spreading code of the transmitter of the successful data packet during the next slot, and users which transmitted a data packet during the current slot are forced to listen for an Ack during the next slot. Given n successful data transmissions in slot t, the number of available transmitters in the next slot is $N_{tx} \leq N - 2n$. When k users transmit in slot $t+1$, the number of busy users is $2n + k \leq N_{bsy} \leq N$ and the number of available receivers is $N_{rx} = N - N_{bsy}$. These parameters allow us to determine the probability that the destination is idle and hence the transition matrix P.

Split-Slot Scheme. An Ack packet is much smaller than a data packet. We can therefore divide slots into two parts, an Ack-slot and a Data-slot. With this design, transmitters are forced to listen for Acks during the next Ack-slot and transmit new data packets or retransmit unsuccessful packets in the following Data-slot. If a receiver receives a data packet without checksum error, the receiver must transmit an Ack packet in the next Ack-slot and can keep quiet, transmit or receive data during the following Data-slot. We denote the length of Ack-slot and Data-slot by L_a and L_d, respectively; L is the ratio of the whole slot to the Ack-slot, i.e., $L = (L_a + L_d) / L_a$. To consider the effect of Ack traffic overhead on the throughput, we modify the normalized throughput by the factor $L_d / (L_a + L_d)$. Three strategies for receiver behavior are possible:

i) Idle after Ack Transmission; In this scheme, we assume that a user cannot switch from the Ack transmission mode to the transmit or receive data mode in the same slot. Therefore successful receivers are allowed to transmit only Acks in the next Ack-slot and must remain idle during the Data-slot. For this scheme, $N_{tx} = N - n$, $N_{bsy} = n + k$, and $N_{rx} = N - N_{bsy}$. *ii) Receive only after Ack Transmission;* In this case, we assume that a user is able to receive a new data packet immediately after having transmitted an Ack. For this scheme, $N_{tx} = N - n$, $N_{bsy} = k$, and $N_{rx} = N - N_{bsy}$. *iii) Transmit/Receive after Ack Transmission;* For this scheme, we assume that a user *can* switch from the Ack transmission mode to either receive mode or transmit mode in the Data-slot and the time to switch modes is negligible. In this case, all N users are allowed to transmit data packets in the next Data-slot. Therefore, $N_{tx} = N$, $N_{bsy} = k$, and $N_{rx} = N - N_{bsy}$.

Our results show that the non split-slot scheme has a maximum throughput of 0.11 (packets/slot/node) whereas the split-slot scheme with receive data only option has a higher maximum throughput of 0.16 (as might be expected). When the Ack-slot overhead is taken into account, our results show that the throughput of the split-slot with receive data option scheme with $L = 4$ (25% overhead) is comparable to that of the non split-slot scheme. We therefore conclude that if Ack overhead is less than 25%, the split-slot scheme has better throughput performance than that of the non split-slot scheme. Extensions to consider a mini slot scheme [3] are currently under investigation.

References

[1] L.G. Roberts, "ALOHA Packet System with and without Slots and Capture," *Comp. Comm. Rev., Vol. 5, No. 2, Apr. 1975, pp. 28-42.*

[2] A.R.K. Sastry, "Effect of Acknowledgement traffic on the performance of Slotted ALOHA-Code Division Multiple Access Systems," *IEEE Trans. on Comm., Vol. COM-32, No. 11, Nov. 1984.*

[3] E.S. Sousa and J.A. Silvester, "A Spreading Code Protocol for a Distributed Spread Spectrum Packet Radio Network," *Proc. of GLOBECOM 84.*

[4] F. A. Tobagi and L. Kleinrock, "The Effect of Acknowledgement Traffic on the Capacity of Packet-Switched Radio Channels". *IEEE Trans. on Comm., Vol. COM-26, No. 6, 1978.*

AN ITERATIVE MODEL TO ANALYZE MULTIHOP PACKET RADIO NETWORKS[1]

SZU-LIN SU and VICTOR O.K. LI
Communication Sciences Institute
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-0272

ABSTRACT

A performance model of a slotted multihop Packet Radio Network is presented. The message generation processes at the nodes are assumed to be independent Bernoulli processes. A node may transmit or receive, but not both at the same time. The transmission queue at each node is modeled as a discrete-time Markov chain. The dependencies of the transmission queues at different nodes are assumed to be characterized by certain parameters of these Markov chains. An iterative procedure is developed to solve for these parameters. Several different transmission protocols will be considered. We have used our model to analyze both narrowband Slotted ALOHA and code division multiple access spread spectrum networks. In the example networks that we have considered, the analytic results match the simulation results very closely.

1. INTRODUCTION

Since the proposal of the ALOHA multiple access protocol by Abramson [1] in 1970, many protocols have been developed for multiple access in packet radio networks. These include Carrier Sense Multiple Access (CSMA) [5], Busy-Tone Multiple Access (BTMA) [16], the Tree algorithm [4], etc. Numerous studies have been made on the performance of these protocols. However, most of them assumed that the system is fully-connected and that each node has at most one packet in the buffer, i.e., buffer size is one; or, if nodes with infinite buffering capabilities are allowed, then only the service delay instead of the total system delay has been analyzed under the assumption of heavy traffic. Boorstyn et al. [2], used the fact [3] that a reversible stationary continuous time Markov chain possesses a product form solution for the steady state probability distribution to find the maximum throughput (without delay analysis) of multihop network under the CSMA protocol. Saadawi and Ephremides [10] developed an iterative procedure for analyzing the delay/throughput performance of a fully-connected slotted ALOHA network with buffered users by a 2-Markov chain model, consisting of the User Markov chain and the System Markov chain.

The difficulty in the analysis of a multihop packet radio network with buffered nodes stems from the dependency between the various queues in the network. This dependency exists since the departure process and part of the arrival process (the transit packets) of a queue are affected by the transmission processes of its one-hop and two-hop neighbors.

To solve this mathematically intractable problem, we usually have two alternatives:

---

[1] This research is supported in part by the Army Research Office under contract No DAAG29-85-K-0016

One is to perform an accurate analysis on some special network configurations in order to understand the true behavior of these simple multihop systems [12], [15]. The other is to propose some (strong) assumptions to make the analysis tractable for general networks and if these assumptions lead to analytic results with a fair degree of accuracy when compared with the real system or simulation results, then we have a good reason to accept them. Segall [11] first proposed the independence assumption for multihop slotted ALOHA network, namely, each queue in the network behaves independently of other queues. He found this simplified model to be a good approximation in heavy traffic. Lee and Silvester [6], [7] adopted an idea similar to that of Saadawi and Ephremides' 2-Markov chain model to correct the overly pessimistic result of Segall's model for light traffic. They defined a parameter, called the busyness probability, for each queue. This parameter is assumed to contain all the effects of this queue on other queues in the system. So each queue in the network is not completely independent of other queues but is affected by their "busyness probabilities" and an iterative procedure is necessary to get the solution.

In this paper, we extend Lee and Silvester's model by distinguishing between newly transmitted and retransmitted packets. We are thus able to analyze a richer set of transmission protocols. We apply our model to the narrowband slotted ALOHA network and the slotted CDMA network. We compare the analytic results of our model with simulation for some example networks and find a good match between them.

2. THE NETWORK MODEL

We consider a network consisting of geographically dispersed nodes. Each node has a radio unit (transceiver) with equal power. So each node can transmit or receive, but not both simultaneously; and if node i can hear node j, i.e., node i is a neighbor of node j, then node j can hear node i. Each node may serve as a repeater of a transit packet which is generated at another node and forward it towards its destination. The time axis is divided into slots and the network nodes are synchronized so that fixed length packets will be sent at the beginning of slots. We assume the acknowledgement for a successful transmission is instantaneous and error-free. The traffic matrix $[t_{ij}]$ and the routing strategy are given so that we can calculate ϕ_{ij} and Λ_i. t_{ij} is the required traffic from node i to node j. ϕ_{ij} is defined to be the fraction of the packets arriving at node i, including both the newly generated packets and the transit packets, which is directed to node j, a neighbor of node i, i.e., ϕ_{ij} is the probability that a packet in the queueing buffer of node i is directed to node j. Λ_i is the total required transit traffic in each slot at node i. Note that $\sum_{j \varepsilon N_i} \phi_{ij} = 1$, where N_i denotes the set of neighbors of node i. The arrivals of new packets from outside the network at different nodes, are assumed to be independent Bernoulli processes. The arrival rate at node i λ_i per slot. We also assume that the total transit traffic at node i (i.e., traffic from i's neighbors) has a Bernoulli distribution with rate γ_i per slot and only when node i is not transmitting in that slot will this transit transmission be successful. So, in the steady state, conservation of flow dictates that $\gamma_i(1-G_i) = \Lambda_i$, where G_i is the average transmission rate per slot at node i.

Figure 1 shows a typical radio unit at a node, say node i. It includes a transceiver, an infinite queueing buffer for newly arrived packets (we can relax this "infinite" assumption easily in the analysis) and a one-unit backlogged buffer for the backlogged packet which is waiting for retransmission. We say that the node is in the idle mode if both buffers are empty, that the node is in the backlogged mode if there is a packet in the backlogged buffer and that the node is in the active mode if the backlogged buffer is empty but the queueing buffer is not empty. After an unsuccessful transmission, the failed packet will be sent to the backlogged buffer and the node enters the backlogged mode. In the backlogged mode, the transceiver will attempt to transmit the backlogged packet in the next slot with probability Pr(i). After a successful transmission, the successful packet will leave the buffer and the node will enter the active mode (nonempty queueing buffer) or the idle mode (empty queueing buffer). In the active mode, the transceiver will attempt to transmit the packet at the head of the queueing buffer in the next slot with probability P(i). Note that when the transceiver is not transmitting, it is in receiving mode.

In this paper, we will analyze three transmission protocols: (1) DFT (Delayed-First Transmission) - A packet which has arrived at the head of the queueing buffer will be transmitted in a slot with probability P(i), provided the node in not backlogged. Therefore, the first transmission attempt of any packet suffers a geometric delay with mean 1/P(i). (2) IFT (Immediate-First Transmission) - When the node is idle, a newly arrived packet will be transmitted immediately in the next slot. Otherwise, the first transmission attempt suffers a delay just as in the DFT protocol. (3) CT (Continuous Transmission) - The CT protocol is actually a special case of both the DFT and the IFT protocols in which we choose P(i) = 1 for all i. It means that after a successful transmission, the packet at the head of the queueing buffer will be transmitted immediately. In all three protocols, the backlogged packets have higher priority than those in the queueing buffer.

In this paper we study the narrowband (non-spread spectrum) slotted ALOHA network and the slotted CDMA network. The major difference in their analyses lies in the computation of the success probability of a transmission. The latter has a more complicated computation and we shall use the result developed in [14]. Since we assume the channel is noise free, for the narrowband system, a transmission will be successful if and only if: i) the intended receiver is in the receiving mode in that slot and ii) no other neighbor of the receiver transmits in the same slot. For the CDMA system, we use frequency-hopping spread spectrum communication technology and receiver-based pseudonoise (PN) code protocol. In addition, we make use of error-correcting codes (such as the Reed-Solomon code used in the examples) to correct the errors due to multiuser and jamming interference.

3. PERFORMANCE ANALYSIS

Although there is a strong dependency between the queues at neighboring nodes, we assume that the effects of other nodes on one particular node are totally encapsulated in some node parameters. We may then consider each queue separately and derive an expression for the average queue length L_i, and hence the average delay. For convenience, we will drop the indices for individual nodes in the following figures and analyses, i.e., we will use P, Pr, λ and γ to replace P(i), Pr(i), λ_i and γ_i, etc.

3.1 Analysis of the DFT Protocol

Figure 2 is the discrete-time Markov chain of a queue under the DFT protocol. A state is denoted by (n_1,n_2), where n_1 indicates the total number of packets in the queueing buffer and backlogged buffer, and n_2 is 0 if the node is not backlogged and is 1 if it is backlogged. The parameters used for the transition probabilities (per slot) in this chain are defined (having dropped the index i) as follows:

λ = Prob(a new packet is generated)
γ = Prob(a transit packet attempts to arrive at this node)
P = Prob(transmission | the node is in the active mode)
Pr = Prob(transmission | the node is in the backlogged mode)
α = Prob(success of the transmission | the node is in the active mode and it transmits)
β = Prob(success of the transmission | the node is in the backlogged mode and it transmits)
and $\bar{X}$ = 1−X, where X represents one of the above parameters.

We note that P and Pr are chosen by the designer, λ is dictated by the traffic requirements of the system, and γ, as we have mentioned in the previous section, is calculated from the equation $\gamma(1-G) = \Lambda$. G and γ will be computed by an iterative procedure. We shall discuss the derivation of α and β later on.

Let $\pi_{i,j} \triangleq$ steady-state probability of state (i,j)

$$U(z) \triangleq \sum_{n=1}^{\infty} \pi_{n,0} z^n \qquad V(z) \triangleq \sum_{n=1}^{\infty} \pi_{n,1} z^n$$

and $S(z) \triangleq \pi_{0,0} + U(z) + V(z)$. Therefore, $S(1) = 1$.

The average queue length in this node is $\bar{L} = S'(1) = \left.\frac{dS(z)}{dz}\right|_{z=1}$. The average transmission probability in a slot of this node is $G = U(1) \bullet P + V(1) \bullet Pr$. We solve this chain [13] and the results are

$$\pi_{0,0} = 1 - \frac{(\lambda+\gamma)(Pr\beta+P\bar{\alpha})}{PPr(\beta+\beta\gamma+\bar{\alpha}\gamma)} \qquad U(1) = \frac{Pr\beta}{P\bar{\alpha}+Pr\beta}(1-\pi_{0,0})$$

$$V(1) = \frac{P\bar{\alpha}}{P\bar{\alpha}+Pr\beta}(1-\pi_{0,0}) \qquad \bar{L} = \frac{H(1) \bullet F'(1) - H'(1) \bullet F(1)}{PPr(\beta+\beta\gamma+\bar{\alpha}\gamma) \bullet H(1)}$$

where

$$H(1) = (P\bar{P}r\bar{\alpha}+\bar{P}Pr\beta)(\lambda+\gamma)+PPr(\bar{\alpha}\lambda-\beta\bar{\lambda})$$

$$H'(1) = -\bar{P}\bar{P}r(\lambda+\gamma)^2 -(\bar{P}Pr\bar{\beta}\lambda-P\bar{P}r\alpha\bar{\lambda})(\lambda+\gamma)+(P\bar{P}r\bar{\alpha}+\bar{P}Pr\beta)\lambda\gamma+PPr\beta\bar{\lambda}\bar{\lambda}+PPr\alpha\lambda\bar{\lambda}$$

$$F(1) = (Pr\beta + P\bar{\alpha})(\lambda+\gamma)$$

$$F'(1) = (Pr\beta + P\bar{\alpha})\lambda\gamma - \bar{P}r(\lambda+\gamma)^2 + (P\bar{\alpha} - Pr\bar{\beta})\lambda(\lambda+\gamma)$$

Note that if we let P = Pr and combine states (i,0) and (i,1) into one state (i) for all i $\geq$ 0 (using β to represent the total effect of α and β), then our model degenerates into the model described in [6]. We have verified that, in this case, the results we derived are exactly the same.

3.2 Calculation of the Probabilities of Successful Transmissions

As we have mentioned before, from the given traffic matrix and routing matrix, we can get the flow matrix for the load on each link and calculate ϕ_{ij}. We define

$\psi_{ij} \triangleq$ Prob(the packet in the backlogged buffer is directed to node j, a neighbor of node i) and $P(s \mid i \to j) \triangleq$ Prob(the transmission from i to j is successful $\mid$ i sends a packet to j) Then we have

$$\alpha_i = \sum_{j \in N_i} \phi_{ij} \cdot P(s \mid i \to j), \qquad \beta_i = \sum_{j \in N_i} \psi_{ij} \cdot P(s \mid i \to j)$$

Since the average number of retransmission attempts that a packet from i to j needs is equal to $1/P(s \mid i \to j) - 1$; therefore,

$$\psi_{ij} = \frac{\phi_{ij}(1/P(s \mid i \to j) - 1)}{\sum_{n \in N_i} \phi_{in}(1/P(s \mid i \to n) - 1)}$$

For the narrowband slotted ALOHA network, a transmission from i to j is successful if: i) node j is in receiving mode, and ii) no neighbor of node j, except node i, is transmitting in that slot. So,
$P(s \mid i \to j) = (1 - G_j) \prod_{\ell \in N_j \setminus \{i\}} (1 - G_\ell)$.

Note that for the narrowband network, we assume that G, the average transmission probability of a node, is the only factor which will affect other nodes (queues). These G's will be found by an iterative procedure.

For the CDMA network we adopt the result of [14]:

$$P(s \mid i \to j) = (1 - G_j) \sum_{\forall D_{ij}^1, D_{ij}^2} \frac{P_c(J + |D_{ij}^1| + |D_{ij}^3|)}{1 + |D_{ij}^1|} C_{1+|D_{ij}^1|} \cdot \prod_{n \in D_{ij}^1} g_{nj} \cdot \prod_{k \in D_{ij}^2} (1 - G_k) \cdot \prod_{m \in D_{ij}^3} (G_m - g_{mj})$$

where D_{ij}^1, D_{ij}^2 and D_{ij}^3 are disjoint sets and $D_{ij}^1 \cup D_{ij}^2 \cup D_{ij}^3 = N_j \setminus \{i\}$.

$P_c(m+J)$ is the packet success probability with m other simultaneous transmissions and the jamming interference is being modeled as J simultaneous transmissions. C_M is the capture probability when there are M transmissions destined to the same receiver with the same PN code in the same slot, and

$$\begin{aligned} g_{nj} &\triangleq \text{Prob(node n transmits a packet to node j in a slot)} \\ &= U_n(1) \cdot P(n) \cdot \phi_{nj} + V_n(1) \cdot Pr(n) \cdot \psi_{nj} \end{aligned}$$

We assume that both G's and g's will affect the other queues. So a more complicated iterative procedure is needed for the CDMA network.

3.3 Iterative Procedures

λ_i, Λ_i and ϕ_{ij} can be calculated from the given traffic matrix and routing matrix for all i and (i,j) pairs. P(i) and Pr(i) are given by the communication system designer. For the CDMA network, we are given the functions $P_c(m+J)$ and C_M.

3.3.1 Narrowband Slotted ALOHA System

1. $n \leftarrow 0$, $G_i^{(0)} = 0.5\,(P(i) + Pr(i))\ \forall\, i$. (The superscript of G_i denotes the iteration number.)
2. Compute $P(s \mid i \to j)\ \ \forall\, (i,j)$.
3. Compute $\psi_{ij}\ \ \forall\, (i,j)$.
4. Compute α_i and $\beta_i\ \ \forall\, i$.

5. Compute $\gamma_i = \dfrac{\Lambda_i}{1-G_i^{(n)}}$ ∀ i.

6. Compute $\pi_{0,0_i}$, $U_i(1)$ and $V_i(1)$ ∀ i. (The subscript i identifies the node.)

7. $n \leftarrow n+1$ and $G_i^{(n)} = U_i(1)P(i) + V_i(1)Pr(i)$ ∀ i.

8. If $|G_i^{(n)} - G_i^{(n-1)}| > \rho$ for any i, go to step 2. (ρ is the tolerance.)

9. Compute $\bar{L}_i$.

10. Check that the results satisfy (i) $PPr(\beta+\beta\gamma+\bar{\alpha}\gamma) \geq (\lambda+\gamma)(Pr\beta+P\bar{\alpha})$, and (ii) $\bar{L}_i > 0$.
    If conditions are not satisfied, stop. Given traffic requirements cannot be satisfied.

3.3.2 CDMA System

1. $n \leftarrow 0$, $G_i^{(0)} = 0.5(P(i) + Pr(i))$, i.e., $U_i(1) = 0.5$ and $V_i(1) = 0.5$ ∀ i.

2. $m \leftarrow 0$, $\psi_{ij}^{(0)} = \phi_{ij}$ ∀ (i,j). (The superscript on ψ_{ij} denotes the iteration number.)

3. Compute $g_{ij} = U_i(1)\, P(i)\phi_{ij} + V_i(1)Pr(i)\, \psi_{ij}^{(m)}$ ∀ (i,j).

4. Compute $P(s \mid i \rightarrow j)$ ∀ (i,j).

5. $m \leftarrow m+1$, compute $\psi_{ij}^{(m)}$ ∀ (i,j).

6. If $|\psi_{ij}^{(m)} - \psi_{ij}^{(m-1)}| > \delta$ for any (i,j) pair, go to step 3. (δ is the tolerance.)

7. Compute α_i and β_i ∀ i.

8. Compute $\gamma_i = \dfrac{\Lambda_i}{1-G_i^{(n)}}$ ∀ i.

9. Compute $\pi_{0,0_i}$, $U_i(1)$ and $V_i(1)$ ∀ i.

10. $n \leftarrow n+1$ and $G_i^{(n)} = U_i(1)\, P(i) + V_i(1)\, Pr(i)$ ∀ i.

11. If $|G_i^{(n)} - G_i^{(n-1)}| > \rho$ for any i, go to step 3. (ρ is the tolerance.)

12. Compute $\bar{L}_i$.

13. Check the necessary criteria for convergence.

We find that these iterations converge very quickly to within a tolerance of 10^{-4}
We can now find_the network average delay, which is given by Little's formula [8] as:

Delay _= $(\Sigma L_i)/(\Sigma \lambda_i)$

where ΣL_i is the total number of packets in the network and $\Sigma \lambda_i$ is the total network throughput.

3.4 Analysis of the IFT Protocol

The IFT protocol is almost the same as the DFT protocol except for the treatment of the packets which arrive at an idle node. So the discrete-time Markov chain of this protocol is the same as that of the DFT protocol except for the boundary states. Figure 3 shows the Markov chain of the IFT protocol in which we have two special states, (1,3) and (2,3), to denote that there are one and two, respectively, packets in the queueing buffer currently and one of them needs immediate transmission. Note that in this figure, the transition probabilities corresponding to the dotted lines are the same as those in Figure 2, so we only indicate the transition probabilities for states (1,3) and (2,3).

Define $S(z) \triangleq \pi_{0,0} + \pi_{1,3}z + \pi_{2,3}z^2 + U(z) + V(z)$

where $U(z) \triangleq \sum_{k\geq 1} \pi_{k,0}z^k \qquad V(z) \triangleq \sum_{k\geq 1} \pi_{k,1}z^k$

and $\pi_{i,j} \triangleq$ steady-state probability of the state (i,j)

Therefore, $S(1) = 1$ and $\bar{L} = S'(1) = \left.\frac{dS(z)}{dz}\right|_{z=1}$

The average transmission probability in a slot becomes

$$G = \pi_{1,3} + \pi_{2,3} + U(1)\bullet P + V(1)\bullet Pr = \pi_{0,0}(1-\lambda\ \gamma) + U(1)\bullet P + V(1)\bullet Pr$$

The results are [13]:

$$\pi_{0,0} = \frac{H(1)}{Pr[(\bar{\alpha}+\beta)\gamma+\beta][\bar{P}(1-\bar{\lambda}\bar{\gamma})-P]}$$

$$U(1) = \frac{T(1)}{H(1)}\bullet\pi_{0,0}, \qquad V(1) = \frac{Y(1)}{H(1)}\bullet\pi_{0,0} = \frac{\bar{\alpha}(\lambda+\gamma)}{Pr[(\bar{\alpha}+\beta)\gamma+\beta]}, \text{ and}$$

$$\bar{L} = [\lambda+\gamma+\frac{(T'(1)+Y'(1))H(1)-(T(1)+Y(1))H'(1)}{H(1)\bullet H(1)}]\bullet\pi_{0,0}$$

where H(1) and H'(1) are defined in Sec. 3.1

$$T(1) = -[\overline{Pr}\alpha(\lambda+\gamma)(1-\bar{\lambda}\bar{\gamma}) + Pr(\bar{\alpha}+\beta)\lambda(1-\bar{\lambda}\bar{\gamma}) + Pr\beta\lambda\gamma]$$

$$Y(1) = \bar{\alpha}[\bar{P}(1-\bar{\lambda}\bar{\gamma})-P](\lambda+\gamma)$$

$$T'(1) = \alpha\lambda(1-\bar{\lambda}\bar{\gamma}+\gamma)(\lambda+\bar{P}r\gamma) - Pr\beta\lambda^2(2\gamma+1-\bar{\lambda}\bar{\gamma}) - \overline{Pr}\alpha\lambda\gamma(1-\bar{\lambda}\bar{\gamma})$$

$$Y'(1) = \bar{P}\bar{\alpha}(3\lambda^2+6\lambda\gamma+\gamma^2-\lambda^2\bar{\lambda}\bar{\gamma}-2\lambda\bar{\lambda}\gamma\bar{\gamma}) - \bar{\alpha}\lambda(\lambda+2\gamma)$$

So, we can use the same iterative procedures in Section 3.3 except for a slight modification of the equations for G_i and g_{ij}. The formula for G is given above and $g_{ij} = (U_i(1)\bullet P(i) + \pi_{1,3_i} + \pi_{2,3_i})\bullet\phi_{ij} + V_i(1)\bullet Pr(i)\bullet\psi_{ij}$.

4. NUMERICAL EXAMPLES

We now study an example network and analyze it with our model. The network has uniform traffic requirements and uses the static shortest-path routing strategies. For the CDMA (frequency-hopping spread spectrum) system, we assume that the transmitter hops over 200 frequency slots and a (32,16) Reed-Solomon code with available side information is used. Moreover, each packet contains only one codeword and the transmitters transmit packets with one symbol per hop. We use the results in [9] to calculate $P_c(J+m)$, the probability of correct packet reception under multiuser interference. We also assume $C_M = 1$ for all $M \geq 1$. Note that the results have not been normalized by the bandwidth used and by the code rate.

Our network consists of ten nodes. The network topology, routing matrix and flow matrix are shown in Figure 4. Here we choose Pr of each node to be proportional to the successful throughput required at that node and assign them as follows: Pr(1)=0.255, Pr(2)=0.315, Pr(3)=0.225, Pr(4)=0.285, Pr(5)=0.270, Pr(6)=0.135, Pr(7)=0.495, Pr(8)=0.240, Pr(9)=0.135, and Pr(10)=0.315. For the DFT and IFT protocols, we consider two cases with P = Pr and P = 2Pr, respectively. The results are shown in Figures 5, 6, and 7. We have also obtained simulation results and found that the analytic results match the simulation results closely.

5. CONCLUSIONS AND FURTHER RESEARCH

We have developed a model for analyzing multihop packet radio networks with buffered nodes for three different transmission protocols. We apply it to two different communication systems, namely, the narrowband slotted ALOHA-type network and the slotted CDMA network. We apply the model on several simple network configurations (one of them is included as an example) and find that the analytic results closely match the simulation results. We also observe that the delay/throughput performance of the CT protocol is the best and the DFT protocol is the worst of the three protocols. We believe that the performance of the IFT protocol is always better than that of the DFT protocol because they behave exactly the same for heavy traffic, and for light traffic, the DFT protocol will suffer an unnecessary delay for the first transmission attempt. However, it is unclear that the CT protocol will be the best over all cases. Since the IFT protocol has two degrees of freedom, namely, P and Pr, instead of one degree of freedom, Pr, for the CT protocol, it may turn out that for some properly designed parameters, the IFT protocol will have a better performance than the CT protocol. The optimization problem for these protocols for different communications systems will be an interesting research topic.

References

1. Abramson, N. The ALOHA System - Another Alternative for Computer Communication. AFIPS Conf. Proc., Fall Joint Computer Conference, 1970, pp. 281-185.

2. Boorstyn, R.R., Kershenbaum, A. Throughput Analysis of Multihop Packet Radio Networks. Proc. IEEE ICC, June, 1980, pp. 13.6.1-13.6.6.

3. Brazio, J.M., and Tobagi, F.A. Theoretical Results in Throughput Analysis of Multihop Packet Radio Networks. Proc. IEEE ICC, June, 1984, pp. 448-455.

4. Capetanakis, J.I. "Tree Algorithms for Packet Broadcasting Channels." *IEEE Trans. on Information Theory IT-25* (Sept. 1979), 505-515.

5. Kleinrock, L. and Tobagi, F.A. "Packet Switching in Radio Channels: Part I - Carrier Sense Multiple-Access Modes and their Throughput-Delay Characteristics." *IEEE Trans. on Communications COM-23* (Dec. 1975), 1400-1416.

6. Lee, I. and Silvester, J.A. An Iterative Scheme for Performance Modelling of Slotted ALOHA Packet Radio Networks. Proc. IEEE ICC, June, 1982, pp. 6G.1-6G.6.

7. **Lee, I.** *On the Computation of Packet Radio Performance Parameters*. Ph.D. Th., Department of Electrical Engineering, University of Southern California, 1982.

8. Little, J.D.C. "A Proof of the Queueing Formula L = λW." *Operations Research 9* (1961), 383-387.

9. Pursley, M.B. Throughput of Frequency-Hopped Spread-Spectrum Communications for Packet Radio Networks. Proceedings of 1983 Conference on Information Science and Systems, The John Hopkins University, March, 1983.

10. Saadawi, T.N., and Ephremides, A. "Analysis, Stability and Optimization of Slotted ALOHA with a finite number of Buffered users." *IEEE Trans. on Automatic Control AC-26* (June 1981), 680-689.

11. Segall, A. The Modeling of Networks with Radio Links as Applied to C^3 System. Proc. Second MIT/ONR Workshop on Distributed Communication and Decision Problem, 1979, Monterey, California.

12. Sidi, M. and Segall, A. "A Three-Node Packet Radio Network." *IEEE Trans. on Communications COM-32* (Dec. 1984), 1336-1339.

13. Su, S.L. *Performance Analysis of Spread Spectrum Networks*. Ph.D. Th., Department of Electrical Engineering, University of Southern California, 1985. (Also available as Tech. Rept. CSI-85-09-01, Communication Sciences Institute, Department of Electrical Engineering, University of Southern California)

14. Su, S.L., and Li, V.O.K. Performance Analysis of A Slotted Code Division Multiple Access (CDMA) Network. Proc. IEEE MILCOM, October, 1984, pp. 23.3.1-23.3.5.

15. Tobagi, F.A. "Analysis of a Two-Hop Centralized Packet Radio Network - Part I : Slotted ALOHA." *IEEE Trans. on Communications COM-28* (Feb. 1980), 196-207.

16. Tobagi, F.A., and Kleinrock, L. "Packet Switching in Radio Channels: Part II--The Hidden Terminal Problem in Carrier Sense Multiple Access and the Busy Tone Solution." *IEEE Trans. on Communications COM-23* (Dec. 1975), 1417-1433.

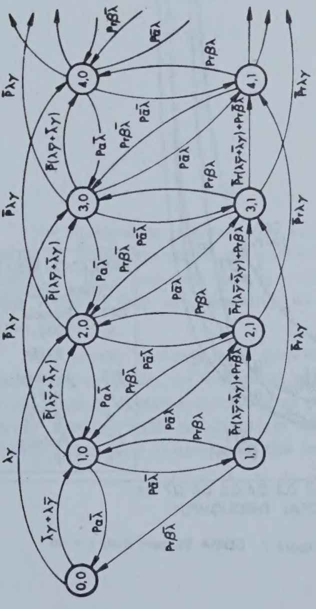


Figure 2. Discrete-time Markov Model for a Radio Unit under the DFT Protocol.

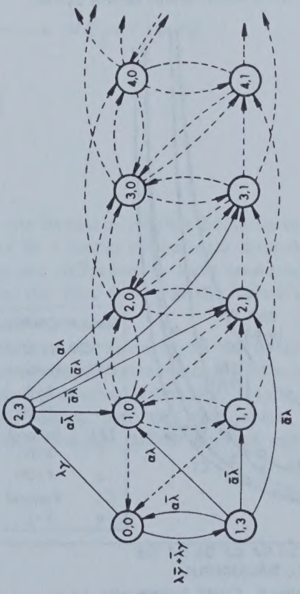


Figure 3. Discrete-time Markov Model for a Radio Unit under the IFT Protocol.

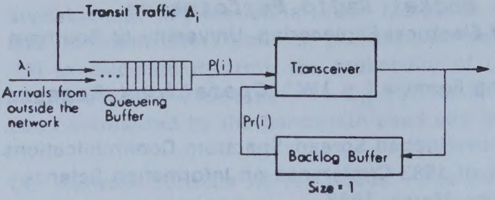
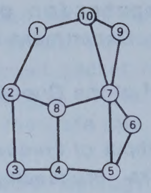

Transit Traffic Δ_i
λ_i
Arrivals from outside the network
Queueing Buffer
P(i)
Transceiver
Pr(i)
Backlog Buffer
Size = 1

Figure 1. Radio Unit of Node i.


1
2
3
4
5
6
7
8
9
10

(a)

$$[r_{ij}] = \begin{bmatrix} - & 2 & 2 & 2 & 10 & 10 & 10 & 2 & 10 & 10 \\ 1 & - & 3 & 3 & 3 & 8 & 8 & 8 & 1 & 1 \\ 2 & 2 & - & 4 & 4 & 4 & 4 & 4 & 2 & 2 \\ 3 & 3 & 3 & - & 5 & 5 & 5 & 8 & 8 & 8 \\ 7 & 4 & 4 & 4 & - & 6 & 7 & 4 & 7 & 7 \\ 7 & 7 & 5 & 5 & 5 & - & 7 & 7 & 7 & 7 \\ 10 & 8 & 8 & 5 & 5 & 6 & - & 8 & 9 & 10 \\ 2 & 2 & 2 & 4 & 4 & 7 & 7 & - & 7 & 7 \\ 10 & 10 & 10 & 7 & 7 & 7 & 7 & 7 & - & 10 \\ 1 & 1 & 1 & 7 & 7 & 7 & 7 & 7 & 9 & - \end{bmatrix}$$

(b)

$$[f_{ij}] = \sigma \begin{bmatrix} - & 8 & - & - & - & - & - & - & - & 9 \\ 8 & - & 9 & - & - & - & - & 4 & - & - \\ - & 7 & - & 8 & - & - & - & - & - & - \\ - & - & 6 & - & 8 & - & - & 5 & - & - \\ - & - & - & 9 & - & 3 & 6 & - & - & - \\ - & - & - & - & 3 & - & 6 & - & - & - \\ - & - & - & - & 7 & 6 & - & 7 & 5 & 8 \\ - & 6 & - & 2 & - & - & 8 & - & - & - \\ - & - & - & - & - & - & 5 & - & - & 4 \\ 9 & - & - & - & - & - & 8 & - & 4 & - \end{bmatrix}$$

(c)

Figure 4. Example 2: (a) Network, (b) Routing Matrix, (c) Flow Matrix.

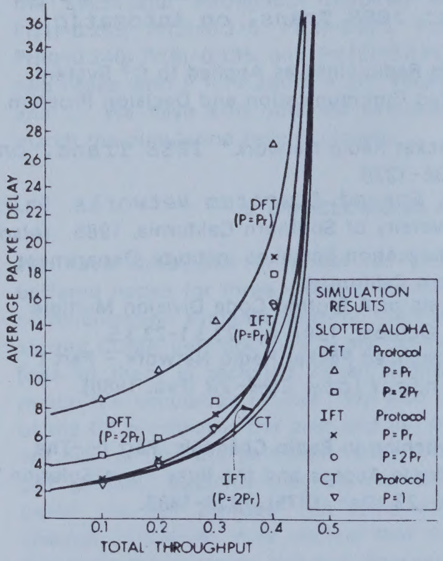


Figure 5 Narrowband Slotted ALOHA System.

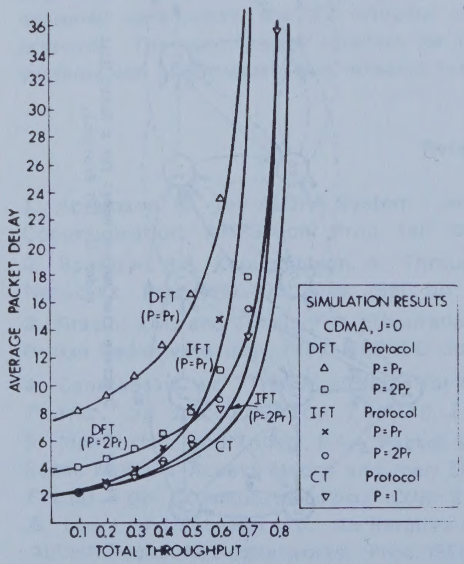


Figure 6 CDMA System with J = 0

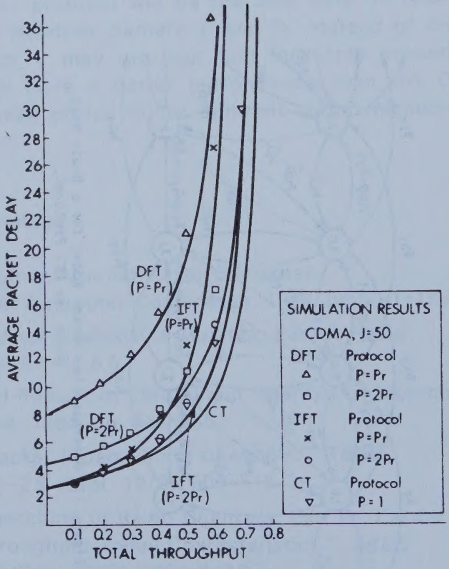


Figure 7. CDMA System with J = 50

AN OPTIMIZATION PROBLEM IN A MULTISERVER QUEUE

by

I.Viniotis A.Ephremides

Univ.of Md,College Park Univ. of Md,College Park

ABSTRACT

We study a two server queueing system,in both continuous and discrete time.We assume **different** service distributions or **unequal** rates or probabilities of service completion.We assume r-stage Erlangian arrival or service distributions in the continuous time models and geometric service times and Bernoulli arrivals in the discrete time model. The objective is to find the control policy which optimizes a discounted cost criterion. We show that the optimal policy is of threshold type for all the models considered ,i.e. the faster server always serves a customer (whenever possible),while the slower server should serve a customer only when the number of customers in queue exceeds a certain threshold.

I.INTRODUCTION

The basic problem to be considered is shown in figure 1 .

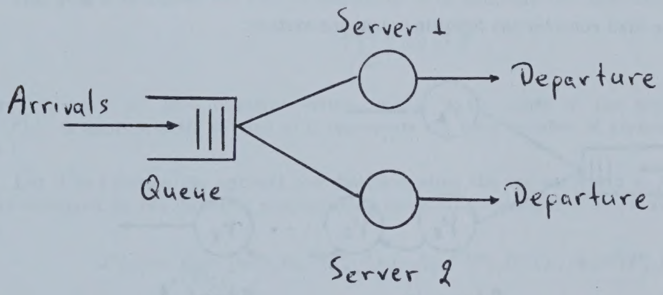


Figure 1

Our motivation comes from examples arising in the dynamic routing problem in computer systems or communication networks. Customers arrive in a single stream and immediately join the queue. It is assumed that the service distributions are **different** or have **unequal** expected values.Without loss of generality we can assume that the first server is the faster one in the second case.

Multiserver systems have provided a focus for much of the research on the optimal control of queueing systems. Lippman [3] proved the existence of optimal stationary policies ,for a more general class of systems,described as semi-Markov decision processes. Larsen [4] considered a multiserver system with nonidentical servers.He assumed Poisson arrivals, exponential servers of different rates, and conjectured the optimality of threshold policies for the average cost criterion.

He provided a detailed analysis of the performance of this class of policies.Using a dynamic programming argument, Lin and Kumar [2] proved Larsen's conjecture. Walrand [10] gave a somewhat simpler proof of the same result,using a probabilistic argument. Hajek [5] considered the case of two interacting nonidentical servers. His framework is quite general,but somewhat different from ours. He showed optimality of "switching" type policies. In [6] and [7],related multiserver systems under different cost criteria are analyzed. Winston [9] considered a discrete time heterogeneous system with multiple customer classes and preemptive fashion of operation. He showed that an optimal policy will reassign customers from a class with longer service requirements to faster servers.

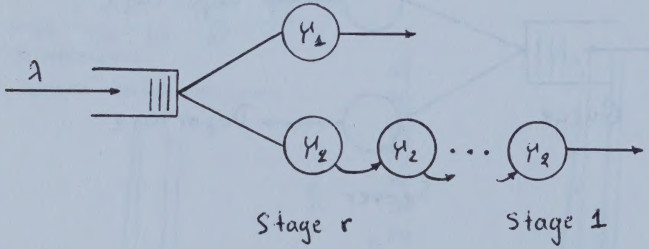
We consider two classes of models of the system shown in figure 1 . In the first class,the time is continuous,while in the second one the time is assumed discrete.The interarrival and service processes are modelled as exponential or r-stage Erlangian processes for the continuous time model,while in the discrete time case,the arrivals form a Bernoulli process and the service times are geometric.

We wish to choose the control actions to minimize a certain (discounted) cost criterion. We show that in all these models the optimal policy of operation is of threshold type,i.e. the faster server should always serve a customer (if there is one waiting),but the slower server should serve a customer only when the number of customers in queue exceeds some threshold value.

In parts II-III we study the continuous time model.In part II we assume that the slower server is r-stage Erlangian,while in part III we assume that the arrival process is r-stage Erlangian.All other processes are assumed exponential. In part IV we study the discrete time model.The arrival process is assumed Bernoulli and the service times geometric.

II.ERLANGIAN SLOWER SERVER

In this part,we shall consider the following queueing system:



Customers arrive in a single Poisson stream of rate λ. They are served by two servers operating in parallel. The first server is exponential with mean service time $\frac{1}{\mu_1}$ while the second server is r-stage Erlangian with mean service time per stage $\frac{1}{\mu_2}$ (or total mean service time $\frac{r}{\mu_2}$).We assume that $\mu_1 > \mu_2$.We assume that a customer cannot enter stage r if another customer is being served in any stage of service.

It is well known (see [1,2] for example),that the continuous time system can be converted to an equivalent discrete time system,by "sampling" the system at appropriate random instants of time.The two systems are equivalent in the sense that the optimal policies for the two formulations are the same (under infinite horizon cost criteria).

Let $x=(x_0,x_1,x_2)$ be the state of the system,just before a sampling instant (i.e. at time t^-),where

x_0 : is the number of customers in queue, $x_0=0,1,2,\cdots$

x_1 :1 *or* 0 ,depending on whether server 1 is busy or not.

x_2 :i , $i=1,\ldots,r$ when server 2 serves a customer in the $i-th$ stage of service.

x_2 :0 when server 2 is idle.

Let $X = Z^+ \times\{0,1\}\times\{0,1,\ldots,r\}$ be the state space of the system.Define operations A, D_1, D_2, mapping X into X as follows:

$A(x_0,x_1,x_2)=(x_0+1,x_1,x_2)$

$D_1(x_0,x_1,x_2)=(x_0,(x_1-1)^+,x_2)$ $\qquad (n^+\equiv max(n,0)\ )$

$D_2(x_0,x_1,x_2)=(x_0,x_1,(x_2-1)^+)$

A represents arrivals,while D_i represent departures from server i . (Note that for the slower server,departure from stage i means transition to the next stage i-1,for i=r,r-1,...,2 ,while departure from stage 1 is the "real" departure of the customer-i.e. completion of his service)

We can similarly define operators P_h , P_1 , P_2 , P_b ,to denote assignment of customer(s) from the queue to the server(s). P_h denotes the action of not forwarding a customer to a server (i.e. holding it),P_1,P_2 denote the actions of forwarding a customer to server 1 or 2 respectively and P_b denotes the action of forwarding two customers to both servers.Note that the customer forwarded to the second server enters the $r-th$ stage of service.

The evolution of the system is influenced by a (state-dependent) control u . Let $U\equiv\{u=(u_0,u_1,u_2) : u_i\in\{h,1,2,b\}\ \ i=0,1,2\}$ be the set of available controls.h denotes holding of a customer,while $1,2,b$ denote forwarding of customer(s) to server 1,server 2 or both servers respectively.Action u_0 is taken whenever an arrival occurs,while action u_i is taken whenever a customer departs from server $i, i=1,2$.

Let $U(x)\equiv\{u\in U\ :\ Ax\in dom\,\{P_{u_0}\}\ ,\ D_i\in dom\,\{P_{u_i}\}\ ,\ i=1,2\ \}$ be the set of admissible controls when the system is in state x .

Our goal is to choose the control actions so as to minimize the discounted cost

$$E\sum_{t=0}^{\infty} |\,x(t)\,|\,\beta^t$$

where $0<\beta<1$ is a discount factor, $x(t)$ is the state of the system at time t and $|\,x(t)\,|$ is appropriately defined so it represents the total number of *customers* in the system at time t.

Let $J^\beta(x)$ denote the optimal cost function,when the system starts at state x . Then $J^\beta(x)$ is characterized by the dynamic programming optimality equation $J^\beta(x)=TJ^\beta(x)$,or more explicitly

$$J^\beta(x)=\min_{u\in U(x)}\{xe+\beta\lambda J^\beta(P_{u_0}Ax)+\beta\mu_1 J^\beta(P_{u_1}D_1x)+\beta\mu_2 J^\beta(P_{u_2}D_2x)\}\qquad\text{(I)}$$

The definition of operator T is obvious from the above equation.

Based on these facts,we can prove lemma I.1,used later in theorem I.2. Lemma I.1 asserts that action P_1 is preferable to action P_2 or P_h ,for certain states x .

Lemma I.1.

i. $J^\beta(P_1x)\leq J^\beta(P_hx)\qquad x\in dom\,\{P_1\}$

ii. $J^\beta(P_1x)\leq J^\beta(P_2x)\qquad x\in dom\,\{P_1\}\cap dom\,\{P_2\}$

Proof

We have to show that (see I)

$$P_1 xe \;+\; \min_{u_0}\beta\lambda J^{\beta}(P_{u_0} A P_1 x) + \min_{u_1}\beta\mu_1 J^{\beta}(P_{u_1} D_1 P_1 x) + \min_{u_2}\beta\mu_2 J^{\beta}(P_{u_2} D_2 P_1 x) \le$$
$$P_h\, xe \;+\; \min_{u_0}\beta\lambda J^{\beta}(P_{u_0} A P_h\, x) + \min_{u_1}\beta\mu_1 J^{\beta}(P_{u_1} D_1 P_h\, x) + \min_{u_2}\beta\mu_2 J^{\beta}(P_{u_2} D_2 P_h\, x) \qquad (A)$$

$$P_1 xe \;+\; \min_{u_0}\beta\lambda J^{\beta}(P_{u_0} A P_1 x) + \min_{u_1}\beta\mu_1 J^{\beta}(P_{u_1} D_1 P_1 x) + \min_{u_2}\beta\mu_2 J^{\beta}(P_{u_2} D_2 P_1 x) \le$$
$$P_2 xe \;+\; \min_{u_0}\beta\lambda J^{\beta}(P_{u_0} A P_2 x) + \min_{u_1}\beta\mu_1 J^{\beta}(P_{u_1} D_1 P_2 x) + \min_{u_2}\beta\mu_2 J^{\beta}(P_{u_2} D_2 P_2 x) \qquad (B)$$

It suffices to show that there exists a nondecreasing function f which satisfies (A),(B) and such that Tf is nondecreasing and satisfies (A) and (B) as well. Inductively then $T^{(n)}$ (operator T applied n times) has the same properties for any n ,and since $J^{\beta} = \lim_{n\to\infty} T^{(n)} f$,J^{β} will have the desired properties (A) and (B) as well.

So we have to show that

I. $Tf\,(P_1 x) \le Tf\,(P_h\, x)$

II. $Tf\,(P_1 x) \le Tf\,(P_2 x)$

III. $Tf\,(x) \ge Tf\,(y)$ for $x \ge y$

Note that there exists at least one such function f ,the identically zero function.

We can prove I,comparing corresponding terms directly. To prove II,we consider the difference $Tf\,(P_1 x)-Tf\,(P_2 x)$ and show that it is nonnegative,using the fact that $\mu_1 > \mu_2$ .For details see [9]. Also,we can easily show monotonicity of $Tf\,(x)$ in each argument, proving thus III.

Theorem I.2 states that the faster server should be kept busy whenever possible.Its proof follows easily from Lemma I.1 (see also [2],[9]).

Theorem I.2: Whenever the faster server is idle,it is optimal to start serving a customer if one is waiting for service.

The consequence of Theorem I.2 is that we can only consider policies which keep the faster server always busy.We can thus reduce the set of admissible controls, without affecting the optimal cost function. That is,we need only decide which of the two states $(k,1,0)$ or $(k-1,1,r)$ is preferable.We prove subsequently that this decision should be made according to a threshold policy.

Lemma I.3 says that any limit point of policies generated by the standard policy iteration algorithm is an optimal policy. For the proof see [2].Lemma I.4 ,the basis of the main result,we show two facts: 1) each iteration of the standard policy improvement algorithm applied to a policy of threshold type again results in a policy of threshold type and 2) the new policy obtained by policy improvement has a threshold at most 1 unit more than the threshold of the original policy.These facts are used in theorem I.5 to show that the optimal policy is of threshold type.

Lemma I.3: Let $\{\pi_n\}$ be the sequence of policies generated by the policy iteration algorithm,and let π_* be a limit point of $\{\pi_n\}$,i.e. $\lim_{k\to\infty} \pi_{n_k}(x) = \pi_*(x)$ for every $x \in X$ .Then π_* is optimal.

Define now a threshold policy with threshold m , denoted by t_m ,as a policy which dispatches a customer to server 2 (when it is idle) if and only if the number of customers in the queue is *strictly* larger than m . Server 1 ,the faster server should be kept busy whenever possible.More precisely,define the operator F_m as

$$F_m(x)=\begin{cases} P_h(x) & \text{if } \{x_0=0\} \text{ or } \{x_1=1 \,,\, x_2\neq 0\} \text{ or } \{x_1=1 \,,\, x_2=0 \,,\, x_0 \leq m\} \\ P_1(x) & \text{if } \{x_1=0 \,,\, x_2\neq 0 \,,\, x_0 \geq 1\} \text{ or } \{x_1=0 \,,\, x_2=0 \,,\, 1 \leq x_0 \leq m+1\} \\ P_2(x) & \text{if } \{x_1=1 \,,\, x_2=0 \,,\, x_0 > m\} \\ P_b(x) & \text{if } \{x_1=0 \,,\, x_2=0 \,,\, x_0 > m+1\} \end{cases}$$

Then a threshold policy t_m , with threshold m (i.e. a policy which supplies a customer to the slower server iff the number of waiting customers exceeds m) ,is defined as

$$t_m(x)=(F_m(Ax),F_m(D_1x),F_m(D_2x))$$

Also

$$T_{t_m}J_{t_m}^{\beta}(x)=J_{t_m}^{\beta}(x)=xe+\beta\lambda J_{t_m}^{\beta}(F_m Ax)+\beta\mu_1 J_{t_m}^{\beta}(F_m D_1x)+\beta\mu_2 J_{t_m}^{\beta}(F_m D_2x)$$

Lemma I.4: For any finite $i \geq 0$, there exists a j , $0 \leq j \leq i+1$ such that $T_{t_j}J_{t_i}^{\beta} = TJ_{t_i}^{\beta}$ .That is,there exists a threshold policy t_j ,which has a threshold not larger than $i+1$,which achieves the minimum cost over all policies.Thus policy iteration on t_i produces a threshold policy again.

Proof

For the proof,we define the cost differences

$$h_k = J_{t_i}^{\beta}(k,1,0)-J_{t_i}^{\beta}(k-1,1,r) \qquad k \geq 1$$

$$h_0 = J_{t_i}^{\beta}(0,1,0)-J_{t_i}^{\beta}(0,0,r)$$

and determine when the differences change sign.If $h_s < 0$ for $s \leq j-1$ and $h_s \geq 0$ for $s \geq j$,then the policy iteration algorithm produces a threshold policy with threshold $j-1$.

Thus we have to examine h_k .Our starting threshold is,by hypothesis i (the threshold policy is t_i).

After some algebra,(see [9] for the details),we can see that for $i \geq 3$, $k > 1$:

$$h_k \geq 0 \qquad k \geq i+2 \tag{5}$$

$$h_k \geq \beta\mu_1 h_{k-1} \qquad k = i+1 \tag{6}$$

$$h_k \geq \frac{\beta\mu_1}{1-\beta+\beta\mu_1+\beta\lambda} h_{k-1} \qquad k = i \tag{7}$$

$$-(1-\beta)h_k+\beta\lambda(h_{k+1}-h_k) \leq \beta\mu_1(h_k-h_{k-1}) \qquad 2 \leq k \leq i-1 \tag{8}$$

Using the above relationships,it's a simple matter to show that $\{h_k\}$ is a nondecreasing sequence and that

$$h_1 < h_2 < \cdots < h_j < 0 < h_{j+1} < \cdots < h_{i+2}$$

for some j between 1 and $i+1$. Thus the policy generated by the policy iteration algorithm (applied on policy t_i) is t_j ,with $1 \leq j \leq i+1$. We can similarly prove that the new policy is t_j ,with $0 \leq j \leq i+1$,for $i=0,1,2$ and $k=1$.Thus the proof of the lemma is complete.

We are in a position now to state the main result:

Theorem I.5: i) The optimal (stationary) policy is of threshold type with a threshold $m^* \leq \infty$.

ii) if $J_{t_i}^{\beta}(x) < J_{t_{i+1}}^{\beta}(x)$ for some $x \in X$ then $m^* \leq i$

Proof

Start policy iteration with policy t_0 .If $T_{t_0}J_{t_0}^{\beta} = TJ_{t_0}^{\beta}$,then t_0 is optimal.If not,then from lemmas I.4 and I.3 it follows that $J_{t_1}^{\beta} \leq J_{t_0}^{\beta}$ with strict inequality for some x X .Again by policy iteration on t_1 , either t_1 is optimal or we shall obtain a new policy t_2 which is better than t_1 and so on. 5.i follows from the fact that policy iteration produces a threshold policy again,when applied to a threshold policy. 5.ii follows from the fact that if we start the policy iteration algorithm with $\pi_0 = t_i$,then $j \leq i+1$.Since the value $i+1$ is by assumption excluded, $j \leq i$ .By repeated application of the algorithm,we see that m^* cannot exceed i .

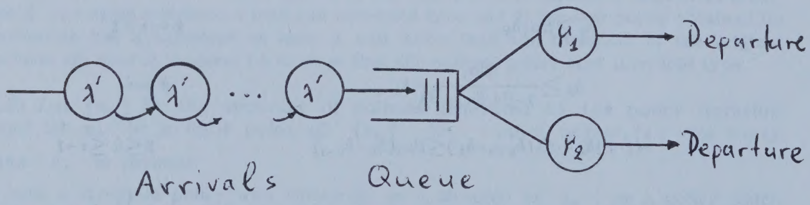
It is easy to see the behavior of the policy iteration algorithm.If policy iteration is started with t_0 ,then each iteration will increase the threshold by one unit at most,until the optimal threshold is reached.Thus if t_j is the optimal policy,it will be reached in exactly j steps of the policy iteration procedure.

Let us summarize the results we have obtained in this part:

We have proved that for the given model (Erlangian slower server),the optimal policy is of threshold type,with an optimal threshold m^* .We have described how policy iteration could give the optimal policy t_j and thus the value j of the optimal threshold m^* ,a significant parameter for implementation purposes.However,this procedure does not seem to be satisfactory,since it requires evaluation of $J_{t_i}^{\beta}(x)$, which is in the best case tedious (if not impossible).

III.ERLANGIAN ARRIVALS

We consider next the following queueing model:



Each customer arrives to the system in r stages. We assume that a customer arrival has been completed (and thus the customer can be forwarded for service) when all r stages have arrived.The time a customer spends in each stage is an exponentially distributed variable with rate λ' ,so the arrival process is r-stage Erlangian,with overall rate $\lambda = \frac{\lambda'}{r}$.The two servers are exponential,with rates $\mu_1 > \mu_2$.A server picks up a bulk of r stages (one customer) at a time.

Let $x = (x_0, x_1, x_2)$ be the state of the system,where

x_0 : is the number of *stages* in queue, $x_0 = 0, 1, \cdots, r, \cdots$

x_i : r *or* 0,depending on whether server i is busy or not.

The state space X is now $X = Z^+ \times \{0, r\} \times \{0, r\}$.The operators A, D_1, D_2 and the customer assignment operators P_1, P_2, P_h, P_b are defined on X as in part II,with some minor notational modifications.

Our goal is again to choose the control actions so as to minimize a similar discounted cost as in part II. $|x(t)|$ will of course be defined differently now, but it still denotes the total number of *customers* in system at time t .

The proof proceeds in the same way as in part II.Lemma II.1, theorem II.2,lemma II.3 and theorem II.5, the main result,are stated and proved in exactly the same way,with some minor notational modifications. There is a major difference in lemma II.4,however,thus we proceed with the outline of its proof.

The threshold policy t_m ,with threshold m is defined in the same way as

$$t_m(x) = (F_m(Ax), F_m(D_1 x), F_m(D_2 x))$$

Operator F_m is of course defined on different domains,to accomodate the fact that the arrivals are now Erlangian and both servers exponential. Note that m is the number of *customers* above which we always feed a customer to server 2.More precisely,

$$F_m(x) = \begin{cases} P_h(x) & \text{if } \{x_0 < r\} \text{ or } \{x_1 = r, x_2 = r\} \text{ or } \{x_1 = r, x_2 = 0, x_0 \leq mr + r - 1\} \\ P_1(x) & \text{if } \{x_1 = 0, x_2 = r, x_0 \geq r\} \text{ or } \{x_1 = 0, x_2 = 0, r \leq x_0 \leq (m+1)r + r - 1\} \\ P_2(x) & \text{if } \{x_1 = r, x_2 = 0, x_0 \geq (m+1)r\} \\ P_b(x) & \text{if } \{x_1 = 0, x_2 = 0, x_0 \geq (m+2)r\} \end{cases}$$

Lemma II.4: For any finite $i \geq 0$,there exists a j , $0 \leq j \leq i+1$,such that $T_{t_j} J_{t_i}^{\beta} = T J_{t_i}^{\beta}$ .That is,there exists a threshold policy t_j ,which has a threshold not larger than $i+1$,which achieves the minimum cost over all policies.So policy iteration on a threshold policy produces a threshold policy again.

Proof

In [9] it is shown that the differences h_k (see part II, for their definition) can be written in this case as:

$$h_k \geq 0 \qquad k \geq ir + 2r \tag{1}$$

$$h_k \geq \beta\mu_1 h_{k-r} \qquad ir + 2r - 1 \geq k \geq ir + r - 1 \tag{2}$$

$$\beta\mu_1 [h_k - h_{k-r}] \geq \beta\lambda [h_{k+1} - h_k] - (1 - \beta + \beta\mu_2) h_k \qquad ir + r - 2 \geq k \geq 2r \tag{3}$$

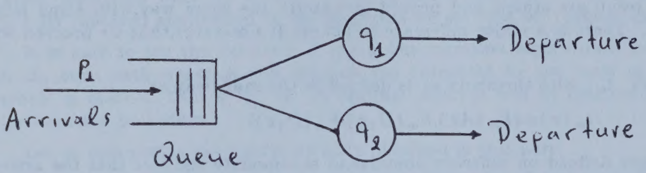
$$h_k \geq \beta\lambda h_{k+1} + \beta\mu_1 [J_{t_i}^{\beta}(k-r, r, 0) - J_{t_i}^{\beta}(k-r, 0, r)] \qquad 2r - 1 \geq k \geq r \tag{4}$$

As in part II,from (1)-(5),we can show that $\{h_k\}$ is monotone and changes sign for some j between 0 and $r(i+1)$, for all k *that are multiples of* r ,i.e. for all k 's that denote *complete* arrivals of customers.Thus the new policy generated by the policy iteration algorithm is of threshold type again,with a threshold not larger than $i+1$.So Theorem II.5 holds true for this model as well.

IV.THE DISCRETE TIME MODEL

For the two models discussed so far,time was assumed continuous.Although this assumption is realistic for analog communication systems,it is not so for digital systems.So naturally one might ask if the previous results apply in a discrete time framework as well.Such a framework seems to be the natural one,to describe the operation of digital computer or communication systems.

In this part,we shall consider the following queueing system:



The system operates in discrete time. Arrivals to the system form a Bernoulli stream,with probability of arrival in a given time slot p_1.The service time at the $i-th$ server is geometrically distributed with parameter q_i,i.e.the probability that the customer completes service in a given time slot is q_i,when the customer is being served by server i. Without loss of generality,we assume that $q_1 > q_2$.

In each time slot,a customer can arrive,or depart from server i. Thus we can have eight possible events $E_i, i=0, \cdots, 7$. The probability of each event,$Pr(E_i)$,can be easily computed from p_1, q_i.Let Q_i be the operator that represents event E_i. Let $x=(x_0,x_1,x_2)$ be the state of the system as usual and $X = Z^+ \times \{0,1\} \times \{0,1\}$ be the state space of the system. Operators P_h, P_1, P_2, P_b have the same meaning as in the previous parts. The set U of available controls is now given by $U \equiv \{u=(u_0,u_1,u_2,u_3,u_4,u_5,u_6,u_7) : u_i \in \{h,1,2,b\}, i=0,\ldots,7\}$, and $U(x) \equiv \{u \in U : Q_i(x) \in dom\, \{P_{u_i}\}\}$ is the set of admissible controls when the system is in state x. The meaning of the controls u_i is explained in part II.

Once again,our goal is to choose the control actions so as to minimize the same discounted cost criterion. $|x(t)| = x_0+x_1+x_2$ is the total number of customers in system at time t.

Note that the dynamic programming equation for this model becomes

$$J^\beta(x) = \min_{u \in U(x)} \{xe + \sum_{i=0}^{7} \beta Pr(E_i) J^\beta(P_{u_i} Q_i x)\}$$

Note also that the threshold policy t_m is now defined as

$$t_m(x) = (F_m(Q_0 x), F_m(Q_1 x), \ldots, F_m(Q_7 x))$$

since we have to provide a control for each one of eight possible events. Operator F_m is now given by

$$F_m(x)=\begin{cases} P_h(x) & \text{if } \{x_0=0\}\ \{x_1=1,x_2=1\}\ \{x_1=1,x_2=0,x_0\le m\} \\ P_1(x) & \text{if } \{x_1=0\ ,\ x_2=1\ ,\ x_0\ge 1\}\ \{x_1=0\ ,\ x_2=0\ ,\ 1\le x_0\le m\} \\ P_2(x) & \text{if } \{x_1=1\ ,\ x_2=0\ ,\ x_0>m\} \\ P_b(x) & \text{if } \{x_1=0\ ,\ ,x_2=0\ ,\ x_0>m+1\} \end{cases}$$

The proof of the corresponding lemmas III.1,III.3,III.4 and theorems III.2,III.5 has a structure similar to that of part II,so for the interests of brevity it is omitted.For details see [9].

VI.CONCLUSIONS

We have considered the problem of controlling a multiserver queueing system with **non-identical** servers. Both continuous and discrete time models were examined.Under the discounted cost criterion,the optimal policy is shown to be of threshold type,for both models. The difficulty in finding the optimal threshold m^* was also discussed.

Several extensions of the results are possible:

(1) One can consider both servers to be Erlangian.

(2) The average cost criterion can be considered also.

(3) The M server case ($M > 2$) can also be studied.

REFERENCES

[1] S.LIPPMAN,"Applying a new device in the optimization of exponential queueing systems" ,Operations Research,vol.23,pp. 687-710,1975

[2] W.LIN-P.R.KUMAR,"Optimal control of a queueing system with two heterogeneous servers",IEEE Trans. on Aut. Control,vol. AC-29,no 8,pp. 696-703 ,1984

[3] S.LIPPMAN,"Semi-Markov decision processes with unbounded rewards",Manag. Sci.,vol. 19,no 7,pp. 717-731,1973

[4] R.L.LARSEN,"Control of multiple exponential servers with application to computer systems",Ph.D. thesis,Univ. of Md,Comp. science dept.,1981

[5] B.HAJEK,"Optimal control of two interacting service stations",IEEE Trans. on Aut. Control,vol. AC-29,no 6,pp. 421-429,1984

[6] C.E.BELL,"Optimal operation of an M/M/2 queue with removable servers", Oper. Res.,vol. 28,no 5,pp. 1189-1204,1980

[7] Y.C.CHOW-W.H.KOHLER,"Models for dynamic load balancing in a heterogeneous multiple processor system",IEEE Trans. on Comp.,vol. C-28,no 5,pp. 354-361, 1979

[8] W.L.WINSTON,"Assignment of customers to servers in a heterogeneous queueing system with switching",Oper. Res.,vol. 25,no 3,pp. 469-483,1977

[9] I.VINIOTIS,"Optimal control of a multiserver system with heterogeneous servers",M.sc. thesis,Univ.of Md.,1985.

[10] J.WALRAND,"A note on 'Optimal control of a queueing system with two heterogeneous servers' ",Systems and Control Letters,vol. 4,pp. 131-134,1984

Optimal Routing and Flow Control of a Network of Parallel Processors with Individual Buffers

Andreas D. Bovopoulos and Aurel A. Lazar

Department of Electrical Engineering
and
Center for Telecommunications Research
Columbia University, New York, NY 10027

ABSTRACT

The problem of routing and flow control of a network of parallel processors is considered. Upon arrival, each packet is routed to an arbitrary number of parallel processors each having its own finite buffer. Every processor can accommodate only one packet at a time, and a processor is accessible if its buffer is not full. To analyze the given class of networks, a closed Markovian queueing model consisting of an appropriate queueing system of parallel processors controlled by a queue with a variable service rate is considered. The optimal state dependent routing and flow control that maximize the average network throughput under an average time delay constraint are given by means of a linear program. The optimal routing is shown to be essentially deterministic and the flow control mechanism of a window type.

I. Introduction

Upon arrival, each packet is routed to an arbitrary number of parallel processors each having its own finite buffer. Every processor can accommodate only one packet at a time, and a processor is accessible if its buffer is not full. This paper addresses the question of finding the optimal routing and flow control of the network described above (see also Fig. 1) under a suitable optimization criterion.

A great deal of work in the literature has investigated this issue under different assumptions and using various optimization criteria. Most of the investigations considered that the arrival packet stream is Poissonian, *i.e.*, only the routing problem has been studied. Winston [14] considered an open queueing system with a finite number of identical exponential servers (without the lower queue of Fig. 1). No jockeying between queues is allowed. Under these assumptions, the intuitive rule of assigning each arrival to the shortest line maximizes, with respect to stochastic order, the discounted number of customers completing service in any given time. Weber [13] extended the results to include random packet service times with a non-decreasing hazard rate. Again, the policy which maximizes the number of customers completing service in a given time assigns each arrival to the shortest queue.

Arriving customers are routed to one of two similar exponential servers (again, without the common buffer in Fig. 1) in the setting studied by Ephremides, Varaiya and Walrand [4]. If the queue lengths at both servers are available, then the optimal decision is to route customers to the shorter queue, whereas, if the queue lengths are not observed, it is best to alternate between queues, provided the initial distributions of the two queue sizes are the same. Furthermore, the optimality of the routing strategies is independent of the statistics of the job arrivals. The optimization criterion used minimizes the expected total time for the completion of service on all jobs arriving before a preassigned time T.

Lin and Kumar [10] studied the problem of optimal control of a queueing system with two heterogeneous servers having a common buffer. The arrivals were Poisson, and the servers were both exponential, but with different mean service times. The optimal policy which minimizes the mean sojourn time (*i.e.*, waiting time in the buffer plus service time) of the customers in the system is of threshold type. The faster server, whenever it is available and whenever the buffer is nonempty is dispatched a customer, but the slower server is dispatched a customer only when, at the instant of dispatching, the number of customers in the buffer exceeds a certain threshold value.

Kumar and Walrand [8] investigated the existence of social and individual optimal policies for a queueing system consisting of a buffer with an arbitrary number of servers. Conditions have been given under which the social and individual policies are equal, and insensitive to the nature of the arrival process.

Recently, Hajek [6] considered the case of two interacting service stations and proved the existence of optimal controls that can be described by a switching curve in the two-dimensional state space. These controls governed routing and service priorities that maximize a linear cost function. Kobayashi and Gerla [7] analyzed a single chain closed queueing network with product form equilibrium state probabilities. Using a combination of the Flow Deviation algorithm and Mean Value Analysis, these authors developed an algorithm for computing the optimal probabilistic routing that maximizes the average throughput of the network.

The network considered in this paper (see also Fig. 1), is modeled as a finite state Markov process [3]. The model, then, consists of an appropriate Markovian queueing system of parallel processors controlled by a queue with a variable service rate. Similar closed queueing systems have been used to model computer communication networks [7], [12]. Our optimization criterion requires finding the state dependent routing and flow control parameters that maximize the average network throughput subject to an average time delay constraint. The choice of this optimization criterion is based on considerations resulting from the design and implementation of flow control protocols in computer communication networks [9] (see also section II). The unknown routing and flow control parameters are state dependent.

The optimal routing and flow control that maximize the average network throughput under an average time delay constraint are explicitly given in terms of a linear program. The structural results derived show that the optimal flow control is a window mechanism, whereas the optimal state dependent routing is essentially deterministic. It must be emphasized that the chosen optimization criterion and especially the introduction of the time delay constraint in the optimization criterion, differentiate this analysis from the above mentioned works. The results presented here hold for an ***arbitrary*** network of parallel processors. Moreover, the methodology developed can be applied to the optimal routing and flow control of Markovian queueing networks in general. The practical limitations of this approach are imposed by the size of the state space.

This paper is organized as follows. In section 2, the queueing network model and the optimization criterion are introduced and the basic notation is defined. In section 3, the optimal routing and flow control of the network of parallel processors is investigated. In Lemma 1 this question is reduced to a nonlinear optimization problem. In the proof of Theorem 1 it is shown that the latter can be transformed into a linear program. Structural properties of the optimal solution are given in Theorem 2. In section 4, the previous analysis is applied to specific networks.

II. The Statement of the Problem

A network of M parallel processors as seen by a source (controller) node together with its acknowledgement path (feedback channel) is schematically depicted in Fig. 1. This queueing model might represent a distributed computer system or a simple computer network. The interpretation of the model is straightforward. Assuming that the routing parameters are the desired ones, the source node (controller) transmits an additional packet whenever a successful transmission is acknowledged by the upper queueing system. Therefore, the service rate of the feedback queue determines the controlled packet arrival rate to the destination node(s). The service rate of the feedback queue λ is bounded by the data capacity of the outbound source link, *i.e.*, the maximum offered load c to the destination node equals the data capacity of the source link. The question addressed in this paper is: what is the set of state dependent optimal routing and flow control parameters that satisfy a suitable optimization criterion?

The performance criterion used here requires maximizing the average network throughput subject to a bounded average time delay constraint (see also [9]). By establishing a bound on the expected time delay one proceeds with considerations which are important in the actual design of computer communication protocols. Time delay is an important system parameter for which bounds are quite natural in the context of making transmissions based upon time outs. Furthermore, the above criterion allows the computation of the maximum throughput for different choices of the maximum permitted average time delay. The closed queueing system is assumed to have N packets. N is a sufficiently large arbitrary integer.

In what follows, the evolution of the queueing network in time will be described by the Markovian stochastic vector $\mathbf{Q}_t = (Q_t^{(1)},\cdots,Q_t^{(M)})$. $Q_t^{(i)}$ is the number of packets waiting or being in service at processor i, $1\le i\le M$.

Let k_i be the number of packets at processor i and k_{M+1} be the number of the packets in the control queue. The state space of the system is

$$E = \left\{\mathbf{k}=(k_1,\cdots,k_M)\,|\,0\le k_i\le N_i,\ \ i=1,2,\cdots,M;\ \sum_{j=1}^{M}k_j\le N,\right\} \quad .$$

For simplicity of notation, the operators $T_{i\cdot}$, $T_{\cdot i}$ and $T_{ij} : E \to E$ for all i, j, $1\le i, j\le M$, defined by

$$T_{\cdot i}(k_1,\cdots,k_i,\cdots,k_M) = (k_1,\cdots,k_i+1,\cdots,k_M) \quad ,$$

$$T_{i\cdot}(k_1,\cdots,k_i,\cdots,k_M) = (k_1,\cdots,k_i-1,\cdots,k_M) \quad ,$$

$$T_{ij}(k_1,\cdots,k_i,\cdots,k_j,\cdots,k_M) = (k_1,\cdots,k_i-1,\cdots,k_j+1,\cdots,k_M) \quad ,$$

are introduced. Following [9], [12], we have the following definitions:

Definition 1. $\lambda=(\lambda_{\mathbf{k}}), \mathbf{k}\varepsilon E$, denotes the control.

Definition 2. The class of controls $\lambda=(\lambda_{\mathbf{k}})$, $\mathbf{k}\varepsilon\, E$, that satisfies the peak constraint

$$0\le \lambda_{\mathbf{k}}\le c \quad , \tag{1}$$

is called admissible.

Note that, the control queue with service rate $\lambda_{\mathbf{k}}$, $0\le \lambda_{\mathbf{k}}\le c$, is equivalent to a control queue with a probabilistic feedback and service rate c. That is, a control queue serving with service rate smaller than its maximum admissible service rate is equivalent to a control queue with feedback, serving at the maximum service rate but feeding back the percentage of packets which are not forwarded into the network. Therefore, the control rate of the control queue can be written:

$$\lambda_{\mathbf{k}} = c(1-r_{\mathbf{k}}^{(\cdot\cdot)}) \quad , \tag{2}$$

where $r_{\mathbf{k}}^{(\cdot\cdot)}$ represents the probability that a packet leaving the control server joins the control queue (no transition takes place in the upper system).

The following *routing parameter* at arrival times to the upper system is defined:

$r_{\mathbf{k}}^{(\cdot i)}$ is the probability that a packet leaving the control server joins the common buffer and a packet in the common buffer joins processor i (a transition from state $\mathbf{k}$ to state $T_{\cdot i}\mathbf{k}$ takes place).

Note that, for $\sum_{j=1}^{M} k_j < N$,

$$r_{\mathbf{k}}^{(\cdot\cdot)} + \sum_{j=1}^{M} r_{\mathbf{k}}^{(\cdot j)} 1(k_j < N_j) = 1 \quad . \tag{3}$$

In what follows $r_{\mathbf{k}}$ denotes the set of routing probabilities:

$$r_{\mathbf{k}} = (r_{\mathbf{k}}^{(\cdot i)}) \quad ,$$

for all i, $1 \leq i \leq M$ and $\mathbf{k} \varepsilon E$.

We are now in the position to define the transition rates $q = (q(\mathbf{k}, \mathbf{j}))$, $(\mathbf{k}, \mathbf{j}) \varepsilon E \times E$ of the Markov process $\mathbf{Q}_t$. These are:

$$q(\mathbf{k}, T_{\cdot i}\mathbf{k}) = c r_{\mathbf{k}}^{(\cdot i)} 1(k_i < N_i) 1(\sum_{j=1}^{M} k_j < N) \quad , \tag{4}$$

$$q(\mathbf{k}, \mathbf{k}) = c r_{\mathbf{k}}^{(\cdot\cdot)} 1(\sum_{j=1}^{M} k_j < N) \quad , \tag{5}$$

and, for all i, $1 \leq i \leq M$,

$$q(\mathbf{k}, T_{i\cdot}\mathbf{k}) = \mu_i 1(k_i > 0) 1(\sum_{j=1}^{M} k_j \leq N) \quad , \tag{6}$$

Let $E\gamma_N$ and $E\tau_N$ denote the average throughput and the average time delay of the upper network of Fig. 1.

Definition 3. The set of routing and flow control parameters $(r_{\mathbf{k}}, \lambda_{\mathbf{k}})$, $\mathbf{k} \varepsilon E$, are said to be optimum over the class of admissible controls for a given upper bound of the average time delay of the network T, $T \varepsilon R_+$, if the maximum throughput

$$F(T) \overset{\Delta}{=} \max_{E\tau_N \leq T} E\gamma_N \tag{7}$$

is achieved.

Note that by introducing the routing probability $r_{\mathbf{k}}^{(\cdot\cdot)}$ the problem of optimal routing and flow control is reduced to a routing problem [5].

III. The Optimal Routing and Flow Control

Our objective in this section is to find the optimal routing and flow control that maximizes the average network throughput under an average time delay constraint as specified by equation (7). This goal is achieved in two steps. In the first step it is shown in Lemma 1 that the question of finding the optimal routing and flow control parameters can be cast as a nonlinear optimization problem. The latter is reduced to a linear program as shown in the proof of Theorem 1. Finally, some structural results of the solution obtained are given in Theorem 2. It is demonstrated that the optimal routing is essentially deterministic and the flow control mechanism is of a window type.

Lemma 1. $(r_{\mathbf{k}},\lambda_{\mathbf{k}})$, $\mathbf{k}\varepsilon E$, *are the optimal routing and flow control parameters if*

$$\sum_{\mathbf{k}\varepsilon E} p(\mathbf{k}) \sum_{i=1}^{M} \mu_i 1(k_i>0) \tag{8}$$

is maximized subject to:

(i) *the global balance equations*

$$\begin{aligned} p(\mathbf{k})\left\{\sum_{i=1}^{M} c r_{\mathbf{k}}^{(\cdot i)} 1(k_i<N_i)\, 1\Big(\sum_{j=1}^{M} k_j<N\Big)+\sum_{i=1}^{M}\mu_i 1(k_i>0)\, 1\Big(\sum_{j=1}^{M} k_j\le N\Big)\right\} = \\ \sum_{i=1}^{M} p(T_{i\cdot}\mathbf{k})\, c r_{T_{i\cdot}\mathbf{k}}^{(\cdot i)} 1(k_i-1<N_i)\, 1(k_i-1\ge 0)\, 1\Big(\sum_{j=1}^{M} k_j-1<N\Big) + \\ \sum_{i=1}^{M} p(T_{\cdot i}\mathbf{k})\mu_i 1(k_i+1\le N_i)\, 1\Big(\sum_{j=1}^{M} k_j+1\le N\Big)\,, \end{aligned} \tag{9}$$

$$\sum_{\mathbf{k}\varepsilon E} p(\mathbf{k}) = 1\,, \tag{10}$$

and $p(\mathbf{k})\ge 0$ *for all* $\mathbf{k}$, $\mathbf{k}\varepsilon E$,

(ii) *the time delay constraint*

$$\sum_{\mathbf{k}\varepsilon E} p(\mathbf{k}) \sum_{i=1}^{M} k_i - T\sum_{\mathbf{k}\varepsilon E} p(\mathbf{k}) \sum_{i=1}^{M} \mu_i 1(k_i>0) \le 0 \quad , \tag{11}$$

(iii) *the capacity constraint* $0\le \lambda_{\mathbf{k}} \le c$ *and routing probabilities* $r_{\mathbf{k}}^{(\cdot i)} \ge 0$, *for all* $\mathbf{k}$, i, $\mathbf{k}\varepsilon E$, $1\le i\le M$, *with*

$$r_{\mathbf{k}}^{(\cdot\cdot)}+\sum_{j=1}^{M} r_{\mathbf{k}}^{(\cdot j)} 1(k_j<N_j) = 1 \quad , \tag{3}$$

for $\sum_{j=1}^{M} k_j<N$.

Proof: Using the transition rates given by the equations (4)-(6), the global balance equations

$$p(\mathbf{k})\sum_{i=1}^{M}\Big\{q(\mathbf{k},T_{\cdot i}\mathbf{k})+q(\mathbf{k},T_{i\cdot}\mathbf{k})\Big\}=\sum_{i=1}^{M} p(T_{i\cdot}\mathbf{k})q(T_{i\cdot}\mathbf{k},\mathbf{k})+\sum_{i=1}^{M} p(T_{\cdot i}\mathbf{k})q(T_{\cdot i}\mathbf{k},\mathbf{k}) \tag{12}$$

can be easily reduced to (9). The throughput of the upper system in Fig. 1 is

$$E\gamma_N=\sum_{i=1}^{M}\mu_i p(k_i>0) \quad ,$$

or equivalently

$$E\gamma_N=\sum_{\mathbf{k}\varepsilon E}p(\mathbf{k})\sum_{i=1}^{M}\mu_i 1(k_i>0) \quad .$$

Since the average time delay amounts to

$$E\tau_N=\frac{\sum_{\mathbf{k}\varepsilon E}p(\mathbf{k})\sum_{i=1}^{M}k_i}{E\gamma_N} \quad ,$$

the condition $E\tau_N \le T$ is equivalent to

$$\sum_{\mathbf{k}\varepsilon E}p(\mathbf{k})\sum_{i=1}^{M}k_i-T\sum_{\mathbf{k}\varepsilon E}p(\mathbf{k})\sum_{i=1}^{M}\mu_i 1(k_i>0)\le 0 \quad . \qquad \blacksquare$$

Theorem 1. *The optimal routing and flow control parameters* $(r_{\mathbf{k}},\lambda_{\mathbf{k}})$, $\mathbf{k}\varepsilon E$, *are given by the equations:*

$$r_{\mathbf{k}}^{(\cdot i)} = \frac{x(\mathbf{k},T_{\cdot i}\mathbf{k})}{x(\mathbf{k},\mathbf{k})+\sum_{j=1}^{M}x(\mathbf{k},T_{\cdot j}\mathbf{k})}\; 1(k_i<N_i)1(\sum_{j=1}^{M}k_j<N) \quad , \tag{13}$$

for all i, $1\le i\le M$,

$$\lambda_{\mathbf{k}} = c(1-\frac{x(\mathbf{k},\mathbf{k})}{x(\mathbf{k},\mathbf{k})+\sum_{j=1}^{M}x(\mathbf{k},T_{\cdot j}\mathbf{k})})\; 1(\sum_{j=1}^{M}k_j<N) \quad , \tag{14}$$

where $x=(x(\mathbf{k},\mathbf{j}))$, $(\mathbf{k},\mathbf{j})\varepsilon E\times E$, *is the solution of the following linear program:*

$$\max\left\{\sum_{\mathbf{k}\varepsilon E}\sum_{i=1}^{M}\left[x(\mathbf{k},\mathbf{k})+\sum_{j=1}^{M}x(\mathbf{k},T_{\cdot j}\mathbf{k})\right]\mu_i 1(k_i>0)1(\sum_{l=0}^{M}k_l<N) + \right. \tag{15}$$

$$\left. \sum_{\mathbf{k}\varepsilon E}\sum_{i=1}^{M}x(\mathbf{k},T_{i\cdot}\mathbf{k})\mu_i 1(\sum_{l=1}^{M}k_l=N)\right\} \quad ,$$

subject to the linear constraints:

$$\sum_{i=1}^{M}x(\mathbf{k},T_{\cdot i}\mathbf{k})c + \sum_{i=1}^{M}x(\mathbf{k},T_{i\cdot}\mathbf{k})\mu_i = \tag{16}$$

$$\sum_{i=1}^{M}x(T_{i\cdot}\mathbf{k},\mathbf{k})c\,1(k_i-1\ge 0)+\sum_{i=1}^{M}x(T_{\cdot i}\mathbf{k},\mathbf{k})\mu_i 1(k_i+1\le N_i) \quad ,$$

$$\sum_{\mathbf{k}\varepsilon E}\left\{x(\mathbf{k},\mathbf{k})+\sum_{i=1}^{M}x(\mathbf{k},T_{\cdot i}\mathbf{k})\right\}1(\sum_{j=1}^{M}k_j<N) + \tag{17}$$

$$\sum_{\mathbf{k}\varepsilon E}x(\mathbf{k},T_{i\cdot}\mathbf{k})1(\sum_{j=1}^{M}k_j=N \text{ and for } \textit{one } i, \textit{ such that } k_i>0\,;\ 1\le i\le M) = 1 \quad ,$$

$$\sum_{\mathbf{k}\varepsilon E}\sum_{i=1}^{M}\left\{x(\mathbf{k},\mathbf{k})+\sum_{l=1}^{M}x(\mathbf{k},T_{\cdot l}\mathbf{k})\right\}k_i 1(\sum_{j=1}^{M}k_j<N) + \tag{18}$$

$$\sum_{\mathbf{k}\varepsilon E}\sum_{i=1}^{M} x(\mathbf{k},T_{i\cdot}\mathbf{k})k_i 1\Big(\sum_{j=1}^{M} k_j=N\Big) -$$
$$T\sum_{\mathbf{k}\varepsilon E}\sum_{i=1}^{M}\left\{x(\mathbf{k},\mathbf{k})+\sum_{j=1}^{M}x(\mathbf{k},T_{\cdot j}\mathbf{k})\right\}\mu_i 1(k_i>0)1\Big(\sum_{l=0}^{M}k_l<N\Big) -$$
$$T\sum_{\mathbf{k}\varepsilon E}\sum_{i=1}^{M}x(\mathbf{k},T_{i\cdot}\mathbf{k})\mu_i 1\Big(\sum_{l=1}^{M}k_l=N\Big)\le 0 \ ,$$

Proof: With each point $\mathbf{k}$, $\mathbf{k}\varepsilon S$, we associate up to $2M+1$ variables $x=(x(\mathbf{k},\mathbf{j}))$, $(\mathbf{k},\mathbf{j})\varepsilon E\times E$, defined by:

$$x(\mathbf{k},\mathbf{k}) \overset{\Delta}{=} p(\mathbf{k})r_{\mathbf{k}}^{(\cdot\cdot)}1\Big(\sum_{j=1}^{M}k_j<N\Big) \quad , \tag{19}$$

and for all i, $1\le i\le M$,

$$x(\mathbf{k},T_{\cdot i}\mathbf{k}) \overset{\Delta}{=} p(\mathbf{k})r_{\mathbf{k}}^{(\cdot i)}1(k_i<N_i)1\Big(\sum_{j=1}^{M}k_j<N\Big) \quad , \tag{20}$$

$$x(\mathbf{k},T_{i\cdot}\mathbf{k}) \overset{\Delta}{=} p(\mathbf{k})1(k_i>0)1\Big(\sum_{j=1}^{M}k_j\le N\Big) \quad . \tag{21}$$

By summing (19) and (20) and using equation (3) we obtain

$$p(\mathbf{k}) = x(\mathbf{k},\mathbf{k})+\sum_{j=1}^{M}x(\mathbf{k},T_{\cdot j}\mathbf{k}) \quad , \tag{22}$$

for $\sum_{j=1}^{M}k_j<N$.

By using the equations (19), (20), (21) and (22) the nonlinear optimization problem stated in Lemma 1, is reduced to the linear program stated in Theorem 1.

Let S be the subset of the state space E, that is accessible under the optimal policy obtained in Theorem 1. Thus

$$S=\{\,\mathbf{k}\varepsilon E \mid p(\mathbf{k})>0\,\} \quad ,$$

with the $p(\mathbf{k})$'s verifying the set of equations (9)-(11). Therefore, S is a convex subset of the state space.

Theorem 2. *The optimal routing is deterministic for all but at most one state in S. The optimal flow control is of a window type (bang-bang control).*

Proof: With every state $\mathbf{k}$, $\mathbf{k}\varepsilon E$, the global balance equation (9) is associated. In addition equation (18) represents a global inequality constraint for the entire system of parallel processors. If $\mathbf{k}\varepsilon S$, we conclude from equation (22) that

$$x(\mathbf{k},\mathbf{k})+\sum_{j=1}^{M}x(\mathbf{k},T_{\cdot j}\mathbf{k}) > 0 \quad , \tag{23}$$

for $\sum_{j=1}^{M}k_j<N$. Therefore, at least one of the elements in the sets $(x(\mathbf{k},\mathbf{k}),x(\mathbf{k},T_{\cdot i}\mathbf{k}))$, $0\le i\le M$ is nonzero for each $\mathbf{k}$, $\mathbf{k}\varepsilon E$. These nonzero elements represent basic variables of an optimal basic feasible solution to the linear program (15)-(18) and their number is equal to the number of constraints given by the equations (9). The time delay constraint allows one more variable of the optimal basic feasible solution to be nonzero.

The existence of an optimal basic feasible solution to the linear program (15)-(18) is guaranteed by the existence of an optimal feasible solution [11]. The existence of the latter for all T, $1/T \leq \max(\mu_1, \mu_2, \cdots, \mu_M)$, can be easily shown based on Lemma 1. This proves, then, that the optimal routing probabilities specified by the equation (13) take the values zero or one except for at most one state in S. For this state one more of the routing probabilities in r_k, or $r_k^{(\cdot\cdot)}$ is nonzero. ▪

IV. Applications

In this section the dependency of the optimal routing and flow control policy on some of the parameters of the queueing network is shown and a comparison with existing results on optimal probabilistic routing is presented.

Fig. 2 shows that the routing switching curve of a network of two parallel processors is a function of the upper bound c. Each of the processors has a buffer of size equal to the number of circulating packets N=6, with $\mu_1 = 4.0$, μ_2=2.0, and, c=1.0 and c=7.0, respectively. The flow control, not shown in the figure, is of a window type.

Finally, in Fig. 3 the performance of a network of three parallel processors with individual buffers is compared using both the optimal state dependent and probabilistic routing. The upper curve is taken when the network follows the optimal routing and flow control policy given in the present paper, whereas the lower curve is based on the results presented by Kobayashi and Gerla [7]. Recall that the analysis in [7] is based on the "classical" product form solutions. As indicated by Fig. 3, the policy outlined in the present paper results in a considerable improvement of the throughput. The pertinent parameters of the network are: μ_1=2.0, μ_2=1.0, μ_3=0.5, c=4.0 and N=5.

V. Conclusions

The optimal routing and flow control of a network of parallel processors has been considered. The model investigated has applications to optimal scheduling of jobs on parallel processors, as well as to routing and flow control in computer communication networks. The problem has been formulated in a general Markovian setting. The routing and flow control parameters have been chosen to be state dependent. Therefore, the equilibrium probabilities, described by the system of global balance equations, do not take a simple analytical form.

Based on design and implementation considerations of communication protocols, the optimization problem investigated calls for maximizing the average network throughput subject to an upper bound on the average network time delay. This has been shown to be equivalent to a nonlinear optimization problem. Subsequently, it has been demonstrated that the latter can be reduced to a linear program.

Structural properties of the optimal solution have been presented. It has been shown that the optimal routing is essentially deterministic and the flow control is a window flow control mechanism. The methodology introduced is very general in its scope, and as such it can be applied to load balancing of distributed computer systems that can be modeled as Markovian queueing networks [2].

Acknowledgement

We would like to thank Zvi Schechner and Cyrus Derman of Columbia University for valuable discussions. The research reported here, was supported in part by the National Science Foundation under Grand # CDR-84-21402 and in part by the Army Research Office under Contract DAAG 29-83-K-0021.

References

[1] Ahmad, H. K. and Lazar, A. A., "The Throughput Time Delay Trade-off of a Birth-Death Process", *Proceedings of the Twentyfirst Annual Allerton Conference on Communication, Control and Computing*, University of Illinois at Urbana-Champaign, Ill, Oct. 8-10, 1983.

[2] Bovopoulos, A.D. and Lazar, A.A., "Optimal Load Balancing for Markovian Queueing Networks", in preparation.

[3] Derman, C., *Finite State Markovian Decision Processes*, Academic Press, New York, 1970.

[4] Ephremides, A., Varaiya, P. and Walrand, J., "A simple Dynamic Routing Problem", *IEEE Transactions on Automatic Control*, Vol. AC-25, No. 4, August 1980, pp. 690-693.

[5] Gallager, R.G. and Golestaani, S.J., "Flow Control and Routing Algorithms for Data Networks", *Proceedings of the Fifth International Conference on Computer Communications*, Atlanta, October 1980, pp. 779-784.

[6] Hajek, B., "Optimal Control of Two Interacting Service Stations", *IEEE Transactions on Automatic Control*, Vol. AC-29, No. 6, June 1984, pp. 491-499.

[7] Kobayashi, H. and Gerla, M., "Optimal Routing in Closed Queueing Networks", *ACM Transactions on Computer Systems*, Vol. 1, No. 4, Nov. 1983, pp. 294-310.

[8] Kumar, P.P. and Walrand, J., "Individual Optimal Routing in Parallel Systems", *Submitted to SIAM Journal on Control and Optimization.*

[9] Lazar, A. A., "Optimal Flow Control of a Class of Queueing Networks in Equilibrium", *IEEE Transactions on Automatic Control*, Vol. AC-28, No. 11, November 1983, pp. 1001-1007.

[10] Lin, W. and Kumar, P.R., "Optimal Control of a Queueing System with Two Heterogeneous Servers", *IEEE Transactions on Automatic Control*, Vol. AC-29, No. 8, August 1984, pp. 696-703.

[11] Luenberger, D. G., *Linear and Nonlinear Programming*, Addison-Wesley, Reading, Mass., 1984.

[12] Robertazzi, T.G. and Lazar, A. A., "On the Modeling and Optimal Flow Control of the Jacksonian Network", *Performance Evaluation*, No. 5, North-Holland, May 1985, pp.29-43.

[13] Weber, R., "On the Optimal Assignment of Customers to Parallel Servers", *J. Appl. Prob.*, No. 15, 1978, pp. 406-413.

[14] Winston, W., "Optimality of the Shortest Line Discipline", *J. Appl. Prob.*, No. 14, 1977, pp. 181-189.

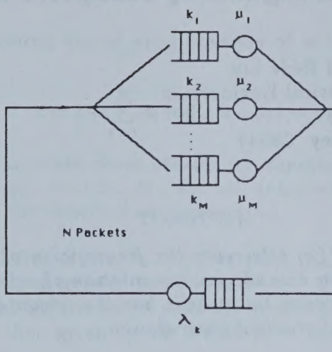

k_1
μ_1
k_2
μ_2
k_M
μ_M
N Packets
λ_k

Figure 1

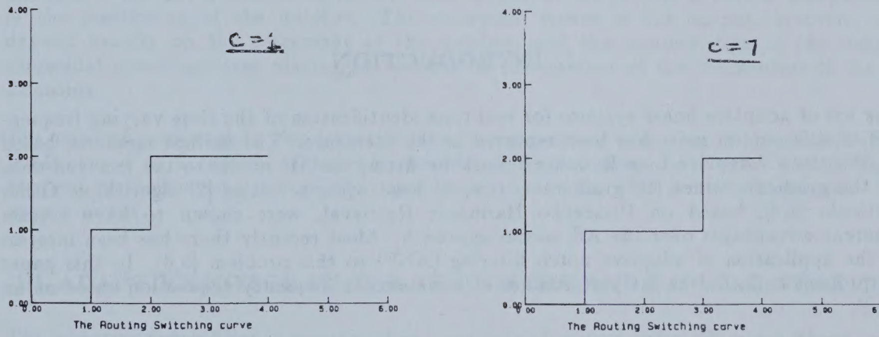


Figure 2

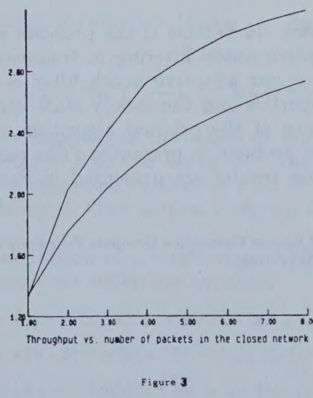


Figure 3

Retrieval of Sinusoidal Signals by Adaptive Notch Filtering

Larry Pearlstein and Bede Liu
Department of Electrical Engineering
Princeton University
Princeton, New Jersey 08544

ABSTRACT

This paper presents a method for retrieving the frequencies of sinusoidal or narrowband signals in additive noise using a cascade implementation of a least mean square Adaptive Notch Filter (ANF). The approach taken here has the advantage that the computational burden is low due to a simplified gradient computation.

We present an analysis of the convergence properties and the steady state error variance of the ANF. Simulation results are presented to illustrate the operation of the ANF and verify the accuracy of the analysis.

I. INTRODUCTION

The use of adaptive linear systems for real time identification of the time varying frequencies of sinusoids in noise has been reported in the literature. The earliest methods, based on Widrow's Adaptive Line Enhancer, work by fitting an AR model to the received data by the gradient ladder [1], gradient lattice, or least squares lattice [2] algorithms. Other methods [3,4], based on Pisarenko Harmonic Retrieval, were shown to have certain inherent advantages over the AR model approach. Most recently there has been interest in the application of adaptive notch filtering (ANF) to this problem [5,6]. In this paper we propose and analyze the performance of a system for frequency acquisition based on an ANF.

Our method has certain distinct advantages over those mentioned above. Perhaps the most significant of these is the low computational burden, which is due to three factors. Firstly, there are the minimal number of parameters to adapt, one for each frequency being tracked. Secondly, a greatly simplified gradient computation is obtained by implementing the adaptive filter in cascade form. Finally, the frequency estimates are directly obtained as simple functions of the adapted parameters, without the complex post-processing required by the other methods. We present results for both the convergence rate and the steady state error variance. These results are confirmed by computer simulations.

This paper is organized as follows. In Section II the problem is formally stated and a brief explanation of the role of adaptive notch filtering in frequency acquisition is given. The structure and implementation of our adaptive notch filter is presented in Section III. In Section IV the convergence properties and the steady state error of the ANF are analyzed. It is shown that the convergence of the original algorithm can be slow, and a modified algorithm which overcomes this problem is presented. The case of several sinusoids is discussed in Section V. Simulation results are presented in Section VI and conclusions are given in Section VII.

This research was supported by a National Science Foundation Graduate Fellowship and the Air Force Office of Scientific Research under Grant AF-AFOSR-81-0186.

II. PROBLEM DESCRIPTION AND ADAPTIVE NOTCH FILTERING

We assume that the incoming signal $x(n)$ consists of a sum of M sinusoids in additive noise,

$$x(n) = \sum_{i=1}^{M} A_i \cos(\omega_i n + \psi_i) + v(n) \tag{1}$$

where $v(n)$ is a zero mean wide sense stationary random process with spectral density $\Phi_v(z)$. The A_i, ω_i, and ψ_i, $i=1,2,\ldots,M$, are all unknown. It is desired to estimate or track $\omega_1, \omega_2, \cdots, \omega_M$ given the observed sequence $x(n)$.

To retrieve the sinusoidal frequencies in (1), We pass the received signal $x(n)$ through a filter whose magnitude characteristic has M notches. The parameters of the filter are adjusted according to some algorithm so that the power at the output is minimized. The sinusoidal frequencies are then given by the locations of the notch frequencies.

The intuitive rationale of adaptive notch filtering is as follows. The power at the output consists of two components, one due to the noise, and the other to the sum of sinusoids. Because of the notch characteristic, the noise power at the output is almost independent of the positioning of the notches. The sinusoidal power at the output, however, will depend heavily on the placement of the notches, and the minimization of the output sinusoidal power involves placing the centers of the notches at the frequencies of the M sinusoids.

The method presented in this paper is based on the above reasoning, where the ANF is implemented as a cascade of tunable second order notch filter sections. The stochastic gradient method is used to tune the notch frequencies of the filter sections to minimize the output power.

III. ADAPTIVE NOTCH FILTER STRUCTURE AND IMPLEMENTATION

The adaptive notch filter is structured as a cascade of second order IIR notch filters, each capable of eliminating one sinusoid. Each filter section has a pair of complex conjugate zeros on the unit circle at angles corresponding to its notch frequency, and a pair of poles with a fixed radius α, near 1, at the same angles.

The transfer function of the overall filter, implemented as a cascade of the sections described above, is given by the product

$$\mathbf{H}_N(z;\mathbf{b}) = \prod_{i=1}^{M} \frac{z^2 + b_i z + 1}{z^2 + \alpha b_i z + \alpha^2}, \tag{2}$$

where $\mathbf{b}$ is the vector of tuning parameters $\{b_1, b_2, \cdots, b_M\}$, and the center frequency of the i^{th} section is given by

$$\Omega_i = \cos^{-1}\left(-\frac{b_i}{2}\right), \qquad -2 \leq b_i \leq 2. \tag{3}$$

The parameter α is fixed, and determines the width of the notch. Its role is analogous to that of the debiasing parameter of Rao and Kung [5]. The choice of a particular value for α involves a tradeoff which is discussed further in Part B of Section VI. For illustration, the pole-zero plot for a single notch filter section is shown in Figure 1.

We now consider the implementation of a single stage ANF, for which the input $x(n)$ and output $y(n)$ are related through the difference equation

$$\begin{aligned} y(n) &= x(n) + b_1(n)x(n-1) + x(n-2) - \alpha b_1(n)y(n-1) - \alpha^2 y(n-2) \\ &= x(n) + x(n-2) - \alpha^2 y(n-2) + b_1(n)[x(n-1) - \alpha y(n-1)]. \end{aligned} \tag{4}$$

Note that we have indicated the dependence on n of the tuning parameter b_1.

To minimize the output power $E\{y^2(n)\}$, the parameter b_1 is adjusted according to the stochastic gradient algorithm

$$b_1(n+1) = b_1(n) - \mu \hat{d}(n) \tag{5}$$

where μ is the step size and $\hat{d}(n)$ is proportional to an estimate of the derivative of the power with respect to $b_1(n)$.

$$\hat{d}(n) = \frac{1}{2}\frac{\mathrm{d}}{\mathrm{d}b_1(n)} y^2(n) = y(n)[x(n-1) - \alpha y(n-1)] \tag{6}$$

Thus

$$b_1(n+1) = b_1(n) - \mu y(n) g(n) \tag{7}$$

where $g(n) \triangleq x(n-1) - \alpha y(n-1)$. A block diagram for the system is shown in Figure 2. Note that $g(n)$ is already available at each step as an intermediate result, and thus requires no additional computation.

We now examine the relationship between $x(n)$ and $g(n)$. Regarding $b_1(n)$ as a constant we have the transfer function

$$H_G(z;b_1) \triangleq \frac{G(z)}{X(z)} = z^{-1}\frac{z^2 - \alpha}{z^2 + \alpha b_1 z + \alpha^2}. \tag{8}$$

This simplification holds when the filter is close to convergence, so that changes in b_1 are small. However, even when the filter is not near convergence, (8) still provides useful insight into the behavior of the ANF. We note that $H_G(z;b)$ describes a system with a sharp band-pass characteristic having its center frequency at the same point as that of the notch filter. Thus $g(n)$ is a line enhanced version of the input signal.

IV. ANF CONVERGENCE AND STEADY STATE ERROR

We consider first the simple case of the one stage ANF processing a single sinusoid in noise. Here the input signal is

$$x(n) = A_1 \cos(\omega_1 n) + v(n) \tag{9}$$

where we have dropped the unknown phase because it is not relevant to the analysis. To facilitate further analysis of the ANF we distinguish between two modes of operation, depending on how far the filter is mistuned from the actual frequency ω_1. We let

$$\delta(n) \triangleq b_1(n) - b^* \tag{10}$$

denote the error in estimating the tuning parameter at time n, where

$$b^* \triangleq -2\cos(\omega_1) \tag{11}$$

is the desired parameter.

A. Notch Center Slightly Mistuned

The following analysis is applicable when the center frequency of the notch filter is tuned sufficiently close to ω_1 that the sinusoid is appreciably attenuated. This is true when $|\delta(n)| < 2(1-\alpha)$. We first develop an approximate expression describing the dependence of $y(n)$ on the sequence $\delta(n)$. Define

$$\begin{aligned} \tilde{y}_j(n) &\triangleq y(n)\,|_{\delta(j)=0}, & \tilde{y}(n) &\triangleq y(n)\,|_{\delta(j)=0,\ all\ j}, \\ \bar{y}_j(n) &\triangleq y(n) - \tilde{y}_j(n), & \bar{y}(n) &\triangleq y(n) - \tilde{y}(n). \end{aligned} \tag{12}$$

From (4), (7), and (12)

$$\bar{y}_j(j) = \delta(j) g(j) \tag{13}$$

The evolution of $\bar{y}_j(n)$ is given by

$$\bar{y}_j(n) = -b^* \alpha \bar{y}_j(n-1) - \alpha^2 \bar{y}_j(n-2) \tag{14}$$

assuming that $b_1(k)$ is close enough to b^* for $k=j,j+1,\cdots,n$ so that second order effects can be ignored. We can write

$$\overline{y}_j(n) = \overline{y}_j(j)h_p(n-j) = \delta(j)g(j)h_p(n-j) \tag{15}$$

where we have defined $h_p(n) = \mathbf{Z}^{-1}[\frac{z^2}{z^2+b^*\alpha z+\alpha^2}]$. Again ignoring second order effects, we have

$$\overline{y}(n) = \sum_{j=0}^{n} \overline{y}_j(n) = \sum_{j=0}^{n} \delta(j)g(j)h_p(n-j) \tag{16}$$

and

$$y(n) = \overline{y}(n)+\tilde{y}(n) = \sum_{j=0}^{n} \delta(j)g(j)h_p(n-j) + v_1(n) \tag{17}$$

where $v_1(n)$ is simply the noise signal passed through $H_N(z;b^*)$ since there is no sinusoidal component in $\tilde{y}(n)$.

Reindexing the convolution of (17) and using it in (6) we obtain

$$\hat{d}(n) = [\sum_{j=0}^{n} \delta(n-j)g(n-j)h_p(j) + v_1(n)]g(n) \tag{18}$$

Now assuming that the filter adapts sufficiently slowly that it can be regarded as time invariant over a short duration we write

$$g(n) = A_1 \mid H_G(e^{j\omega_1};b_1(n)) \mid \cos(\omega_1 n+\phi_G)+w(n) \tag{19}$$

where $\phi_G \triangleq \arg[H_G(e^{j\omega_1};b_1(n))]$ and $w(n)$ is the noise passed through $H_G(z;b_1(n))$. Noting that $H_G(z;b_1(n))$ has a bandpass characteristic with unity gain at center frequency, and ignoring the narrow-band noise term $w(n)$, we substitute (19) into (18) and obtain

$$\hat{d}(n) = \frac{A_1^2}{2}\sum_{j=0}^{n} \delta(n-j)h_p(j)[\cos(2\omega_1 n+2\phi_G-\omega_1 j) + \cos(\omega_1 j)] + A_1\cos(\omega_1 n+\phi_G)v_1(n). \tag{20}$$

Subtracting b^* from both sides of (5) and using (20) we have the following equation for the evolution of the tuning error in the neighborhood of $\delta(n)=0$.

$$\begin{aligned}\delta(n+1) = \delta(n) - \frac{\mu A_1^2}{2}\sum_{j=0}^{n} \delta(n-j)h_p(j)[\cos(2\omega_1 n+2\phi_G-\omega_1 j) + \cos(\omega_1 j)] \\ -\mu A_1\cos(\omega_1 n+\phi_G)v_1(n)\end{aligned} \tag{21}$$

The first cosine term in (21) varies rapidly compared with the evolution of $\delta(n)$, and has little net effect. Thus

$$\delta(n+1) \cong \delta(n) - \frac{\mu A_1^2}{2}\sum_{j=0}^{n} \delta(n-j)h_p(j)\cos(\omega_1 j) - \mu A_1\cos(\omega_1 n+\phi_G)v_1(n) \tag{22}$$

We have found that (22) predicts the final convergence behavior of the single stage ANF quite well. For sufficiently small μ, (22) is stable, so the final convergence of the single stage ANF is exponential.

From (22) it is straightforward to arrive at the following expression for the steady state error variance:

$$\sigma_\delta^2 \triangleq E\{\delta^2(n)\} = \frac{\mu^2 A_1^2}{4}\frac{1}{2\pi j}\int U(z)U(\frac{1}{z})[\Phi_{v_1}(ze^{j\omega_1}) + \Phi_{v_1}(ze^{-j\omega_1})]\frac{dz}{z} \tag{23}$$

where $\Phi_{v_1}(z)$ is the spectral density of $v_1(n)$ and is given by

$$\Phi_{v_1}(z) = \Phi_v(z)H_N(z)H_N(\frac{1}{z}) \tag{24}$$

and

$$U(z) \triangleq \frac{1}{z-1+\frac{\mu A_1^2}{4}[H_p(ze^{j\omega_1})+H_p(ze^{-j\omega_1})]} \tag{25}$$

B. Notch Center Grossly Mistuned

Assuming that the filters adapt sufficiently slowly so that they can be regarded as time-invariant over a short time interval, we write

$$y(n) = A_1 \mid H_N(e^{j\omega_1};b_1(n)) \mid \cos(\omega_1 n + \phi_N) \tag{26a}$$

$$g(n) = A_1 \mid H_G(e^{j\omega_1};b_1(n)) \mid \cos(\omega_1 n + \phi_G) \tag{26b}$$

where ϕ_N and ϕ_G are respectively the arguments of H_N and H_G. We have ignored the noise to clarify the convergence analysis.

Using (26 a) and (26 b) in (8) we obtain

$$\hat{d}(n) = \frac{A_1^2}{2} \mid H_N(e^{j\omega_1};b_1(n)) \mid \ \mid H_G(e^{j\omega_1};b_1(n)) \mid [\cos(2\omega_1 n + \phi_N + \phi_G) + \cos(\phi_N - \phi_G)] \tag{27}$$

Furthermore, it can be shown that

$$\cos(\phi_N - \phi_G) \cong \sin(\omega_1)\mathrm{sgn}[\delta(n)]. \tag{28}$$

When the notch filter is grossly mistuned the sinusoid is passed with very little attenuation, and we can take $\mid H_N(e^{j\omega_1};b_1(n)) \mid \cong 1$. Using the reasoning discussed under Part A we discard the time-varying term in (27), and use (28) to obtain

$$\hat{d}(n) = \frac{A_1^2}{2} \mid H_G(e^{j\omega_1};b_1(n)) \mid \sin(\omega_1)\mathrm{sgn}[\delta(n)]. \tag{29}$$

Inspection of (29) reveals that the rate of convergence is proportional to the level of the sinusoid at the output of the high-Q bandpass filter H_G. Since this filter has a very sharp cutoff, and it is tuned far from the frequency of the sinusoid, this level can be quite small. This implies that the convergence rate in this case can be unacceptably slow. This behavior has been observed in simulations.

In order to overcome the problem of slow convergence, we make the level of $g(n)$ independent of the tuning by using $\mathrm{sgn}[g(n)]$ in place of $g(n)$ in (6). Thus

$$b_1(n+1) = b_1(n) - \mu y(n)\mathrm{sgn}[g(n)]. \tag{30}$$

In this case we find that $\hat{d}(n) = \frac{2}{\pi} A_1 \sin(\omega_1)\mathrm{sgn}[\delta(n)]$.

V. SEVERAL SINUSOIDS

When several sinusoids are to be processed, the appropriate ANF consists of a cascade of the second order sections previously described. In order to adapt the tuning parameter of the i^{th} section according to the stochastic gradient method, the recursion is given by

$$b_i(n+1) = b_i(n) - \mu_i\, y(n)\tilde{g}_i(n) \tag{31}$$

where $\tilde{g}_i(n) \triangleq \sum_{k=0}^{n} g_i(k) f_i(n-k)$, $g_i(n) \triangleq x_i(n) - \alpha y_i(n)$, and $f_i(n)$ is the system function between $y_i(n)$ and $y(n)$. A block diagram of this is shown in Figure 3.

Note that $g_i(n)$ is a narrowband signal centered about Ω_i. From Figure 3 we see that this signal is passed through a cascade of notch filters to obtain $\tilde{g}_i(n)$. Assuming that none of these filters are tuned near Ω_1, we recognize that this additional processing has little effect on $g(n)$. Therefore we do not alter the situation much by eliminating these post filters. The resulting system is shown in Figure 4, which is significantly simpler that the original system of Figure 3.

VI. SIMULATION RESULTS

In this section we present computer simulations of the ANF which corroborate the theoretical results obtained in Section IV and illustrate the behavior of the ANF in two practical situations.

A. Notch Filter Initially Grossly Mistuned

Example 1 - A single sinusoid in white noise is processed by the single-stage ANF. Here we use the standard stochastic gradient algorithm of (5) to update the tuning parameter. The step size, μ, is close to the maximum for which the algorithm is stable. The relatively large value of α causes H_G to have a sharp bandpass characteristic. This is responsible for the slow initial convergence, as discussed in Part B of Section IV. A plot of the ANF center frequency vs. time is shown in Figure 5. The full test conditions of this example and those which follow are indicated on the associated figures.

Example 2 - The conditions here are the same as those of Example 1 with the exception of a smaller step size. The modified algorithm of (30) is used. Note the improved initial convergence rate and steady state error as shown in Figure 6.

B. Choice of α

The following two examples illustrate the compromise involved in choosing a value for α. A smaller value of α leads to smaller steady-state error but α closer to unity produces less bias in the final estimates.

Example 3 - This example shows the effect of α on the bias. The input signal consists of two sinusoids and white noise. The two stage ANF is implemented using the simplified cascade form of Figure 4 and the modified algorithm. The learning curves shown in Figures 7a and 7b were obtained using $\alpha = 0.9$ and $\alpha = 0.7$ respectively, under otherwise identical conditions. Note in these figures that worse final estimation bias accompanied the smaller value of alpha.

Example 4 - The standard algorithm is used. The input to the ANF consisted of a single sinusoid in white noise. Simulations were run under various conditions of μ and α, and the sample variances of $\delta(n)$ were computed. The graph of Figure 8 plots the noise gain vs. μ for 2 values of α. The lines show the theoretical curves obtained using Equation (23) and the points represent data obtained by simulation. Note the excellent agreement between the theoretical results and those obtained by computer simulation. Also, observe in Figure 8 that for a given μ the noise gain is less for smaller α.

C. Multiple Sinusoids

Example 6 - A 3-stage ANF is implemented using the standard algorithm and simplified cascade structure. The input signal contained three sinusoids in white noise. The learning curves for the three stages are plotted simultaneously in Figure 9.

VII. SUMMARY AND CONCLUSIONS

This paper treats the retrieval of sinusoidal signals using a cascade implementation of adaptive notch filtering. The method is simple to implement, and amenable to analysis. Computer simulations were shown to validate the results obtained analytically.

The ANF presented in this paper offers good performance while requiring very low computation when compared with previous approaches. Some areas of interest to be investigated in the future are analysis of the multi-sinusoid case, analysis of the tracking characteristics, performance comparisons with other methods, and analysis of bias.

References

[1] L.J. Griffiths, "Rapid Measurement of Digital Instantaneous Frequency," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-23, pp. 207-222, April 1975

[2] W.S. Hodgkiss, Jr., and J.A. Presley, Jr. "Adaptive Tracking of Multiple Sinusoids Whose Power Levels are Widely Separated," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-29, pp. 710-721, June 1981

[3] P. Thompson, "An Adaptive Spectral Analysis Technique for Unbiased Frequency Estimation in the Presence of White Noise," *Proc. 13th Asilomar Conf. Circ., Syst., Comput.,* pp. 529-533, 1980

[4] V.U. Reddy, et. al. "Least Squares Type Algorithm for Adaptive Implementation of Pisarenko's Harmonic Retrieval Method," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-30, pp. 399-405, June 1982

[5] D. V. Bhaskar Rao and Sun-Yuan Kung, "Adaptive Notch Filtering for the Retrieval of Sinusoids in Noise," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. ASSP-32, pp. 791-802, Aug. 1984

[6] B. Friedlander and J. O. Smith, "Analysis and Performance Evaluation of an Adaptive Notch Filter," *Proceedings ICASSP, 1983,* pp. 1325-1330, 1983

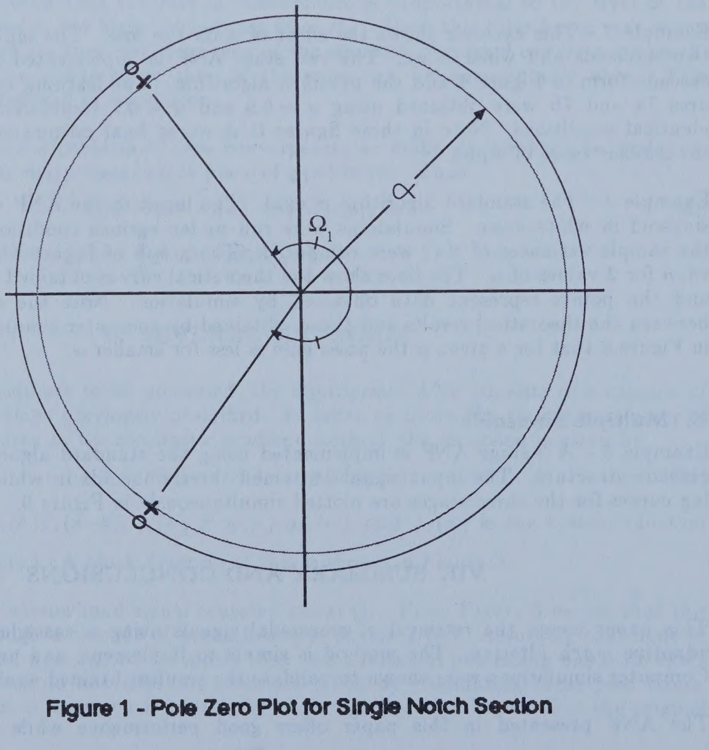


Figure 1 - Pole Zero Plot for Single Notch Section

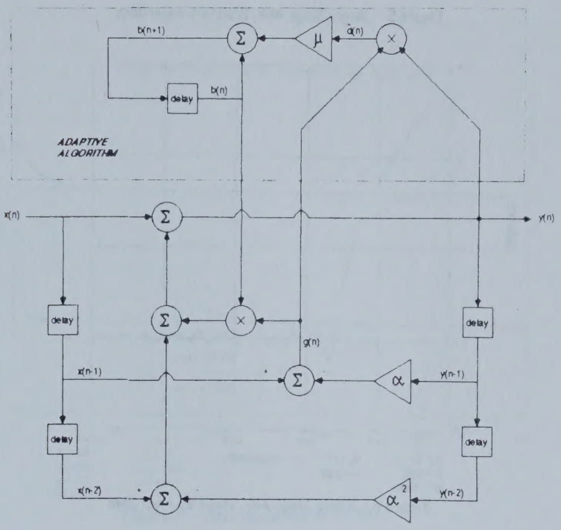

b(n+1)
Σ
μ
â(n)
×
delay
b(n)
ADAPTIVE ALGORITHM
x(n)
y(n)
delay
Σ
×
g(n)
delay
x(n-1)
Σ
α
y(n-1)
delay
delay
x(n-2)
Σ
α²
y(n-2)

FIGURE 2 - ADAPTIVE NOTCH FILTER SIGNAL FLOW DIAGRAM

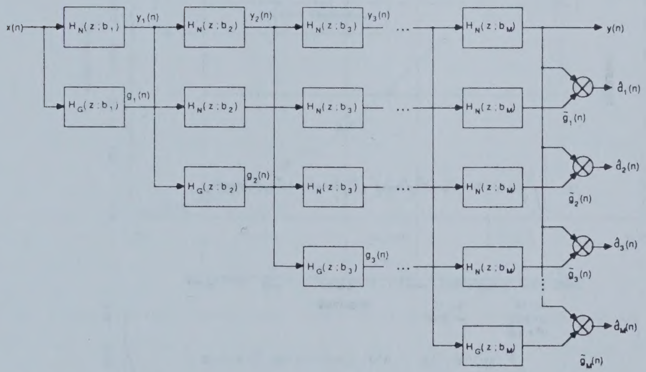

x(n)
H_N(z;b_1)
y_1(n)
H_N(z;b_2)
y_2(n)
H_N(z;b_3)
y_3(n)
H_N(z;b_M)
y(n)
H_G(z;b_1)
g_1(n)
â_1(n)
g̃_1(n)
H_G(z;b_2)
g_2(n)
â_2(n)
g̃_2(n)
H_G(z;b_3)
g_3(n)
â_3(n)
g̃_3(n)
H_G(z;b_M)
â_M(n)
g̃_M(n)

Figure 3 - Standard Cascade Form ANF

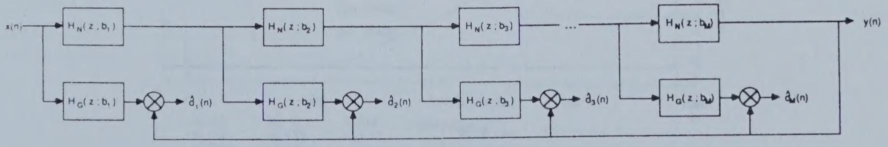

x(n)
H_N(z,b_1)
H_N(z;b_2)
H_N(z;b_3)
H_N(z;b_M)
y(n)
H_G(z;b_1)
â_1(n)
H_G(z,b_2)
â_2(n)
H_G(z,b_3)
â_3(n)
H_G(z;b_M)
â_M(n)

Figure 4 - Simplified Cascade Form ANF

Figure 5 - Single Stage ANF, Standard Algorithm

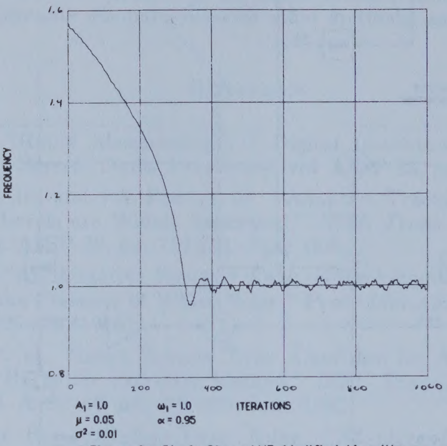


Figure 6 - Single Stage ANF, Modified Algorithm

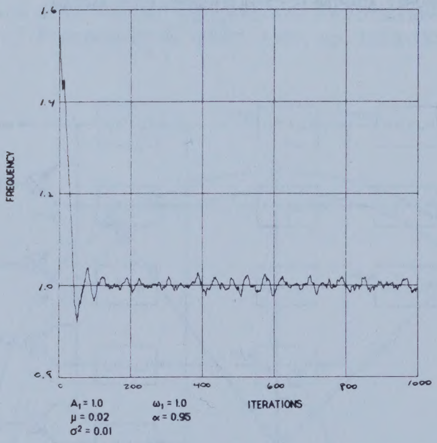


Figure 7a - ANF Learning Curves

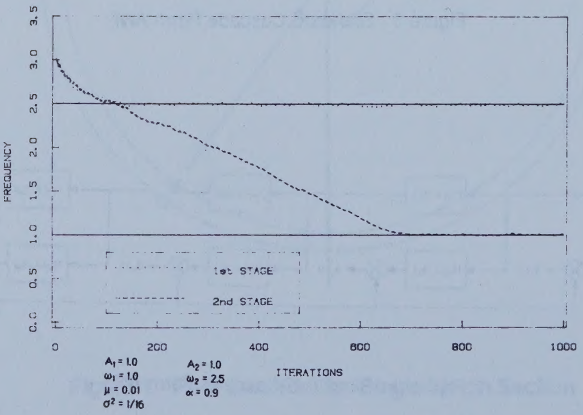


Figure 7b - ANF Learning Curves

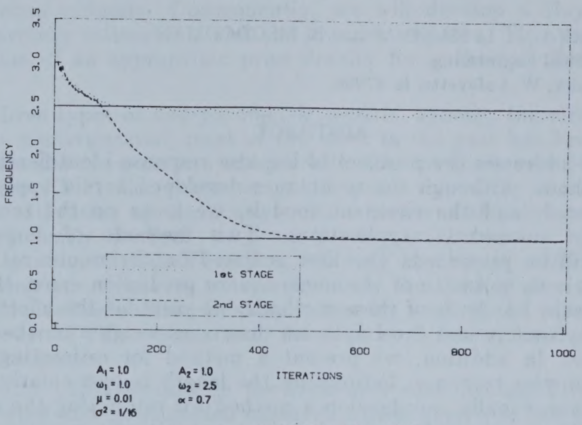


Figure 8 - ANF NOISE GAIN vs. MU

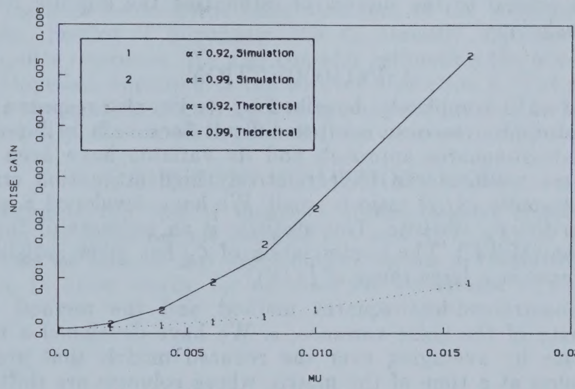
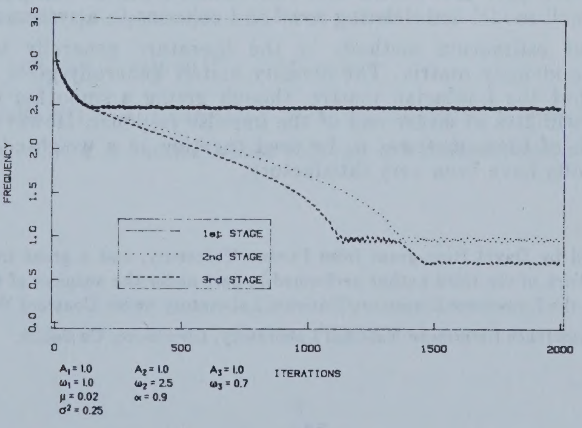


Figure 9 - Typical ANF Learning Curves



IMPULSE RESPONSE IDENTIFICATION WITH DETERMINISTIC INPUTS USING NONPARAMETRIC METHODS.

U. K. BHARGAVA , R. L. KASHYAP and D. M. GOODMAN
School of Electrical Engineering
Purdue University, W. Lafayette, In 47906.

ABSTRACT

This paper addresses the problem of impulse response identification using nonparametric methods. Although the techniques developed herein apply to the truncated, untruncated, and the circulant models, we focus on the truncated model which is useful in certain applications. Two methods of impulse response identification will be presented. The first is based on the minimization of the C_L Statistic, which is an estimate of the mean-square prediction error; the second is a Bayesian approach. For both of these methods, we consider the effects of using both the identity matrix and the Laplacian matrix as weights on the energy in the impulse response. In addition, we present a method for estimating the effective length of the impulse response. Estimating the length is particularly important in the truncated case. Finally, we develop a method for estimating the noise variance at the output. Often, prior information on the noise variance is not available, and a good estimate is crucial to the success of estimating the impulse response with a nonparametric technique.

1. INTRODUCTION

A linear system is completely described by its impulse response. The problem of estimating an impulse response is often difficult because it is ill-posed [1]. In the past constrained-least-squares approach and its variants have been used. As our results show, these methods can have relatively high estimation error, especially when the signal-to-noise (S/N) ratio is small. We have developed a new estimation method based on the C_L statistic. This statistic is an estimate of the mean-square error of prediction (MSEP). The minimization of C_L has given satisfactory impulse response estimates over a large range of (S/N).

Both the constrained-least-squares method and the method based on C_L require an estimate of the noise variance, ρ. We have developed a technique that finds this estimate by averaging over the reduced models that are obtained by deleting one column at a time of the matrix whose columns are shifted versions of the input data. We have also shown that this estimate can be computed quickly by considering a small model and deleting rows and columns in a systematic fashion.

The various estimation methods in the literature generally use either the Laplacian or the identity matrix. The identity matrix generally gives a very oscillatory estimate, but the Laplacian matrix, though giving a smoother estimate, does not give good estimates at either end of the impulse response. However, minimizing C_L permits both of these matrices to be used together in a weighted linear combination. The results have been very satisfactory.

Research supported by David Ross grant from Purdue University, and a grant from the Office of Naval Research. Work of the third author performed in part under the auspices of the U. S. Department of Energy at the Lawerence Livermore National Laboratory under Contract W-7405-Eng-48.

* Currently with Lawrence Livermore National Laboratory, Livermore, Ca 94550.

Estimating ρ by deleting columns is mainly a deterministic approach, and it is difficult to determine the properties of this estimate. Another problem is understanding the effect of the accuracy of an estimate of ρ on the accuracy of the impulse response estimate. Consequently, we will develop a Bayesian procedure that simultaneously estimates ρ and the impulse response. This procedure depends on the selection of an appropriate prior density for ρ, the so-called conjugate density.

Of the three types of non-parametric models, namely, the circulant, the truncated and the non-truncated, most of the work in the past has been with the non-truncated case and to a certain extent with the circulant case [4]. In simulation experiments for EMP, the electro-magnetic pulse caused by a nuclear explosion, and object-identification applications, we have observed that before the desired signals have sufficiently decayed to zero the data may become corrupted by the advent of a spurious signal. Therefore, we may require an estimate of the onset time of the spurious signal so we can truncate the data at the proper time. We solve the problem of estimating this onset time by a scheme in which we recursively change the order of the system, and, for each such order, we find the significant values in the estimate.

Because this paper focuses on the truncated case, Section 2 contains a review of both truncated and untruncated cases. A critical review of the constrained-least-squares method is given in section 3. This method is a fundamental in the development of our methods because the basic structure of the estimator is same for all three methods. Section 4 introduces the C_L statistic and shows its utility for estimating impulse responses. We also consider estimating the noise variance in this section. The Bayesian approach is the subject of section 5. The estimation of the onset time for the spurious signal is developed in section 6. The last three sections contain, respectively, the simulation results, the conclusion, and the references.

2. THE NONPARAMETRIC MODEL

Let us estimate the first M elements of the impulse response sequence h(.) given the observed input-output records x(.) and y(.), both of length N. In the untruncated case we assume that input-output signals are zero before and after the available data. In other words, for $t<O$ and $t>N$ we assume x(.) and y(.) are zero. Then

$$\underline{y} = X_{ut}\underline{h} + \underline{e} \tag{2.1}$$

where,

$$\underline{y} = [\ y(0),y(1),\ldots\ldots y(N-1),0,\ldots\ldots 0\ ]^T \quad N+M-1 \ \ vector$$

$$\underline{h} = [\ h(0),h(1),\ldots\ldots h(M-1)\ ]^T \quad M \ \ vector$$

$$\underline{e} = [\ e(0),e(1),\ldots\ldots e(N+M-1)\ ]^T \quad N+M-1 \ \ noise\ vector \sim Normal(0,\rho I)$$

and X_{ut} is a $N+M-1$ x M matrix whose columns are shifted version of the input signal.

$$X_{ut} = \begin{bmatrix} x(0) & 0 & 0 & . & 0 \\ x(1) & x(0) & 0 & . & . \\ . & x(1) & x(0) & . & . \\ . & . & x(1) & . & 0 \\ x(N-1) & . & . & . & x(0) \\ 0 & x(N-1) & . & . & x(1) \\ . & 0 & x(N-1) & . & . \\ . & . & 0 & . & . \\ 0 & 0 & . & . & x(N-1) \end{bmatrix} \tag{2.2}$$

However, in practice it is often necessary to truncate the data before the system response has completely decayed because the tail of the data is corrupted by a spurious signal that is scattered from objects other than the one under test. If the onset of the spurious signal occurs at N then we can delete the last $M-1$ rows of X_{ut}. However, if the onset time is unknown, it must be treated as a parameter to be estimated. The truncation leads to the following system.

$$\underline{y} = X\underline{h} + \underline{e} \tag{2.3}$$

where, $\underline{y}$, $\underline{h}$ and $\underline{e}$ are now N vectors and X is a N x N matrix.

3. A REVIEW OF THE CONSTRAINED-LEAST-SQUARES METHOD:

Estimating the impulse response from input-output measurements, in the most general sense, is a model fitting problem; thus, a criterion function that measures the goodness of fit is required. When a quadratic function is used, the estimate of the impulse response is often large and oscillatory. However, one can minimize the criterion function subject to the constraint that its weighted Euclidean norm, $||P\hat{\underline{h}}||$, where P is a weighting matrix, be less than some number. Although I and C are the most common weighting matrices, additional prior knowledge may justify the use of some other matrix. In any case, the solution is given by

$$\hat{\underline{h}}_k = \left[X^T X + kP^T P\right]^{-1} X^T \underline{y} \tag{3.1}$$

where k is a nonnegative scalar that must be adjusted to satisfy the energy constraint. This estimator has the well-known ridge structure [5], and k is called the ridge parameter.

The problem with the constraint-least-squares method is that the amount of weighted energy in the impulse response, or, equivalently, the best value for k are seldom available *a priori*. This problem was circumvented in [4] by a " presence of noise generates a minimum amount of uncertainty; Thus, any solution should possess the same uncertainty" argument. In other words, the energy constraint (equivalently, k) should be adjusted such that the energy in the residuals is equal to the expected noise energy. We will call this variant of the constraint-least-squares approach the constrained-residuals method. In the extensive simulations we performed using this method to estimate the impulse response of a dipole we obtained good results for high S/N, however, for low S/N the results deteriorated considerably. The average estimation error was 7 percent for a 33 db S/N ratio but was 89 percent for a 10 db S/N ratio. The amount of error in the 10 db case is clearly unacceptable.

Because the constrained-residuals method fails for low S/N the Bayesian approach and the method based on the C_L-statistic are considered.

4. MINIMIZING ESTIMATED MEAN SQUARE ERROR OF PREDICTION

Because the estimated impulse response may be used for prediction and because a statistic exists that gives an estimate of prediction error, minimizing this statistic is a very attractive approach.

4.1 Mallows' C_L Statistic We fix the structure of the our estimates as shown below. Note especially the third equation where the I and C weighting matrices are used as a linear combination.

$$\hat{\underline{h}}_k = \left[X^T X + k_1 I\right]^{-1} X^T \underline{y} = L_k \underline{y} \tag{4.1.1}$$

$$\hat{\underline{h}}_k = \left[X^T X + k_2 C^T C\right]^{-1} X^T \underline{y} = L_k \underline{y} \tag{4.1.2}$$

$$\hat{\underline{h}}_k = \left[X^T X + k_1 I + k_2 C^T C\right]^{-1} X^T \underline{y} = L_k \underline{y} \tag{4.1.3}$$

The nonnegative scalars k_1 and k_2 are ridge parameters; the method for selecting them will be discussed below. For the sake of brevity we use the symbol L_k for all three cases.

The mean square error of prediction is defined to be

$$MSEP = E\left[\,\| X\underline{h} - X\hat{\underline{h}}_k \|^2\,\right] \tag{4.1.4}$$

and involves the unknown impulse response, $\underline{h}$. Because $\underline{h}$ is unknown, the probability density needed for the evaluation of the expectation is also unknown, and we must use an estimate of MSEP. Several estimates of MSEP are developed in the statistical literature. We obtained very satisfactory results with Mallows' C_L statistic, which is a bias corrected estimate of the MSEP [2]. It has been shown that for large N the C_L statistic is very close to the minimum mean-square estimate of prediction error [3]. The C_L statistic is defined as

$$CL = \frac{RSS(k)}{\rho} - N + 2\,trace(XL_k) \tag{4.1.5}$$

where, $RSS(k) = \| \underline{y} - X\hat{\underline{h}}_k \|^2$ and ρ is the noise variance which can be estimated from the data. We now form a one- or two-dimensional minimization problem where we minimize C_L with respect to k_1 or k_2 or k_1 and k_2 and substitute in (4.1.1), (4.1.2) or (4.1.3) to find the estimated response. The estimation error compared well with that of the constrained-residuals method in the high S/N case, but was smaller in the low S/N case (see Table 1).

4.2 Estimating Noise Variance The usual way for estimating noise variance in a linear system is to find a least squares fit, square the norm of the residual vector, and divide it by the degrees of freedom to get the estimated variance. The degrees of freedom are $N-M-1$. However, when the number of rows are equal or nearly equal to the number of columns this method completely fails or has a very high variance. Because $N = M$ in the truncated case, the number of rows and columns are equal, and a method for estimating ρ based on a reduced model has been formulated.

We define a reduced model of the model of Eq. (2.3) to be

$$\underline{y} = X'\underline{h}' + \underline{\varepsilon} \tag{4.2.1}$$

where X' is N x $(N-1)$ matrix obtained by deleting the i^{th} column of X and $\underline{h}'$ is a $(N-1)$-vector obtained by deleting the i^{th} component of $\underline{h}$. The residual-sum-of-squares from the least-squares fit for this reduced order model is

$$RSS' = \| \underline{y} - X'\hat{\underline{h}}' \|^2 \tag{4.2.2}$$

The random variable $\frac{RSS'}{\rho}$ has a chi squared distribution with one degree of freedom; hence, RSS' has mean ρ and variance 2ρ. To reduce the variance we average RSS' over N, giving the estimate

$$\hat{\rho} = \frac{1}{N} \sum_{i=1}^{N} RSS' \tag{4.2.3}$$

The variance of this estimate is $2\rho/N$. To reduce the computational load, a smaller model can be used. For example X and $\underline{y}$ could be constructed from only every second or third element of x(.) and y(.). If every third element is used, X is $N/3$ x $N/3$; the results from this model were as good as the full N x N model. In our simulations using both the small and full model we observed only three out of sixty cases the estimate fell outside $0.5 > \hat{\rho}/\rho > 2.0$ (see Table 2).

5. A BAYESIAN APPROACH

The Bayesian interpretation which will give Eq. (3.1) as the solution to Eq. (2.3) is well known [6]. It can be easily shown that Eq. (3.1) gives the Bayesian estimate of $\underline{h}$ by assuming a gaussian prior for $\underline{h}$, namely, $N(0,\xi(P^TP)^{-1})$. With this Bayesian approach one can assign a prior density to ρ and have a control on the estimated impulse response. Our development in this section is primarily concerned with this approach for handling ρ.

With the help of the Bayesian formula we write the likelihood function and maximize it simultaneously with respect to $\underline{h}$ and $k = \rho/\xi$. This approach has two advantages. The first is that it avoids the computational load of estimating noise variance separately. The second is that one has a choice of prior densities for ρ.

Consider the model of Eq. (2.3). The conditional probability density of $\underline{y}$ given X and $\underline{h}$ is

$$p(\underline{y}/X,\underline{h}) = \frac{1}{(2\pi\rho)^{N/2}} \exp\left[-\frac{1}{2\,\rho}\,||\underline{y}-X\underline{h}||^2\right] \tag{5.1}$$

Let the prior for $\underline{h}$ be

$$p(\underline{h}) = \frac{1}{(2\pi\xi)^{N/2}|\det(P^TP)|^{1/2}} \exp\left[-\frac{1}{2\xi}\,||P\underline{h}||^2\right] \tag{5.2}$$

where, P is either I or C, and let the probability density for ρ be flat. Then the Bayesian estimates for $\underline{h}$, ρ and ξ are

$$\hat{\underline{h}} = [X^TX + \frac{\rho}{\xi}P^TP]^{-1}X^T\underline{y} \tag{5.3}$$

$$\hat{\rho} = \frac{1}{N}\,||\,\underline{y}-X\hat{\underline{h}}\,||^2 \tag{5.4}$$

$$\hat{\xi} = \frac{||P\hat{\underline{h}}||^2}{N} \tag{5.5}$$

If $\frac{\hat{\rho}}{\hat{\xi}} = \hat{k}$ then we can get

$$\hat{k} - \frac{||\underline{y}-X[X^TX+\hat{k}P^TP]^{-1}X^T\underline{y}||^2}{||P[X^TX+\hat{k}P^TP]^{-1}X^T\underline{y}||^2} = 0 \tag{5.6}$$

We solve (5.6) for $\hat{k}$ and substitute in (5.3) to get the estimated response. Using (5.4) we can also find $\hat{\rho}$, the estimate for noise variance.

Our simulation results show the the estimates for k, and ρ were smaller but varied less than those obtained using the methods discussed previously. The effect of this is seen in the error percentages, *e.g.*, consider the 21 db S/N case using the I matrix in Table 1. The average error is 34 percent for the Bayesian method compared to 20 percent of the constrained-residuals, however, over the 10 runs the error for the Bayesian ranged from 27 to 51 percent, but the error for the constrained-residuals ranged from 9 to 89 percent. These results suggest that Bayesian estimates for the ridge parameter and the noise variance have lower variance but are biased. The bias is thought to be a consequence of an incorrect prior density for the noise variance.

6. ESTIMATING ONSET TIME OF THE SPURIOUS SIGNAL

For estimating N, the onset time of the spurious signal, (see Eq. 2.3) our approach has been to change N and see how the number of significant values of the estimated response change. In other words, we start with a nominal N and find the optimum ridge parameter, k, and also the number of significant values in the estimated impulse response. The ridge parameter is found by minimizing the C_L. We then increase and decrease N in steps and, using the same k, find the change in the number of significant values. That N for which the number of significant values do not change appreciably by increasing N but do change when N is decreased is considered optimum. This testing can be done iteratively with a triangular decomposition technique. An example to applying this method is included in the simulation results in section 7.

7. SIMULATION RESULTS

The numerical results presented here are based on estimating the impulse response of a 0.4 meter dipole antenna. Fig. 1a shows the input sequence, and fig. 1b shows the actual impulse response of dipole. Fig. 1c has the simulated noise free and noisy output sequences ($S/N = 21$ db). The output sequence of fig. 1c is truncated at 55 samples under the assumption that in real world measurements a spurious signal corruption would have taken place. Our experiment consists of 10 runs each for S/N ratios of 33 db, 21 db and 10 db. Our problem is to use one of the simulated noisy output sequences and the input sequence of fig. 1a to estimate the impulse response sequence of fig. 1b. We discuss the simulation results below.

ERROR IN ESTIMATION ($\frac{||h-\hat{h}||}{||h||}$) : Table 1 summarizes the error made by the three methods with the two different weighting matrices with respect to the S/N and the choice of the weighting matrix. It shows the average and the range of the estimation error over the 10 runs. Obviously, as the S/N is lowered, the effectiveness of the methods decreases.

For 33 db S/N the I matrix does better than the C matrix and the constrained-residuals method does best. As the S/N is reduced the C matrix becomes more effective and the constrained-residuals method becomes the least effective. C_L minimization with the simultaneous use of I and C generally does very well.

The basic difference between the effects of the two weighting matrices can be observed in Fig 1d and 1e which show the actual and the estimated responses. Fig. 1d shows that the estimate is highly oscillatory when the I matrix is used. However, with the C matrix the estimate is smoother, but has problems of matching at the ends. We are still searching for a better understanding of and remedies for this phenomenon.

The average error for the Bayesian approach was the greater for all the three S/N ratios, but the error variance was small (see section 5 for an explanation of an example from Table 1).

SENSITIVITY OF ERROR TO THE ACCURACY OF THE ESTIMATES FOR NOISE VARIANCE: The sensitivity of error to the accuracy of the estimate of the noise variance, $\hat{\rho}$, is a crucial issue in this problem. Fig. 1f shows a rather extreme example of this dependence. It shows that for $\hat{\rho} / \rho$ less than 0.5 or greater than 5.0 the error can be very high. Of course, it is more sensitive to $\hat{\rho} < \rho$ because then we approach the unconstrained-least-squares solution.

NOISE VARIANCE: For the reduced-model method there were only three out of sixty cases in which $\hat{\rho}$ fell out of the range of $0.5 < \hat{\rho} / \rho > 2.0$ (see Table 2), which is encouraging when we consider the sensitivity of impulse response estimation error to $\hat{\rho}$ as in Fig. 1f.

The bayesian approach, on the other hand, gave $\hat{\rho}$ consistently smaller than ρ but with very little variation. Once again this is a very attractive feature of this approach and is the reason for the impulse response estimation error being high but low in variance. It is ironic that $\hat{\rho}$ is consistently smaller than ρ because from Fig. 1f this is the bad area for error.

IMPULSE RESPONSE LENGTH ESTIMATION: Table 3 is a work sheet for estimating the impulse response length. In this example the simulated output signal was generated from the impulse response of Fig. 1b which has about 50 significantly nonzero values. We are testing to see if this length of 50 can be estimated from the test of significance based method of section 4.2. From this table it is seen that as the order is increased above 50 the number of significant values in the estimated response do not appreciably increase and thus the impulse response length is correctly estimated as 50.

8. CONCLUSION

In this paper we have considered nonparametric methods for estimating an impulse response. Our emphasis was on EMP and object-identification applications. We discussed the usual constrained-least-squares method as a vehicle for introducing our new methods, namely, the minimization of C_L and a Bayesian approach. We also showed that C_L could be minimized with respect to more than one ridge parameter; this allowed us to use the I and C weighting matrices in tandem. The result was a smoother and more accurate estimate. We have also considered the problem of a spurious signal corrupting the data. Consequently two extra problems had to be solved, namely, estimating the noise variance of a square system and estimating the length of the impulse response. The simulation results have shown good promise for all the methods of this paper.

9. REFERENCES

1. M. P. Ekstrom, *'A spectral characterization of the ill-conditioning in numerical deconvolution'*, IEEE trans. on audio and electroacoustics, Vol. AU-21, NO. 4, August 1973, pg. 344-348.
2. C. L. Mallows, *'Some comments on Cp'*, Technometrics, Vol. 15, No. 4, Nov. 1973, pg. 661-675.
3. O. Bunke, B. Droge, *'Estimators of the mean squared error of prediction in linear regression'*, Technometrics, Vol. 26, No. 2, May 1984, pg. 145-155.
4. B. R. Hunt, *'Biased estimation for nonparametric identification of linear systems'*, Mathematical Biosciences, Vol 10, 1971, pg. 215-237.
5. A. E. Hoerl and R.W.Kennard, *'Ridge regression: biased estimation for nonorthogonal problems,'* Technometrics, Vol.12, pp 55-67, 1970.
6. G.H.Golub, M.Heath, and G.Wahba, *'Generalized cross-validation as a method for choosing a good ridge parameter,'* Technometrics, Vol.21, No.2, pp. 215-223,May, 1979.

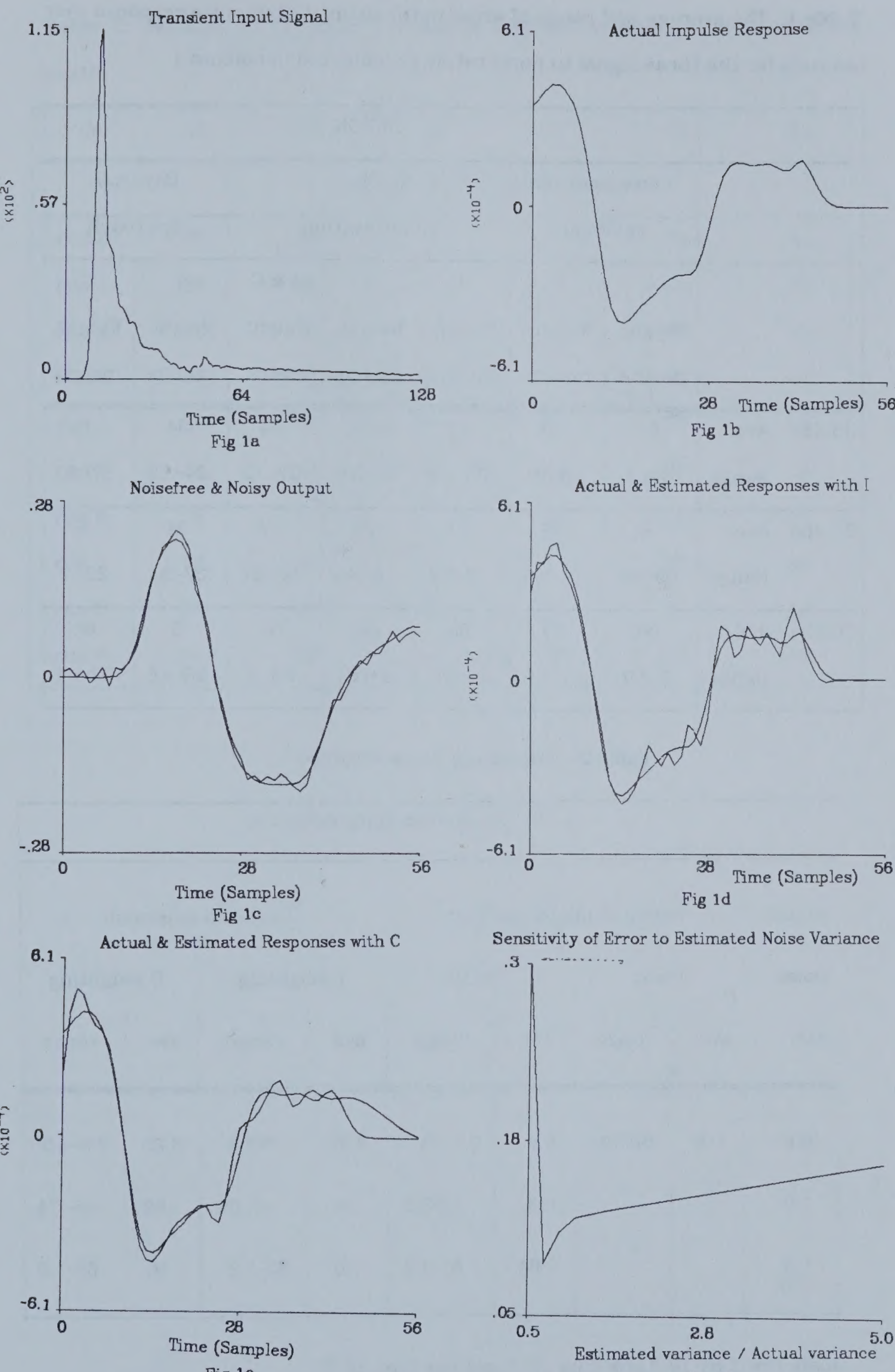


Fig 1a

Fig 1b

Fig 1c

Fig 1d

Fig 1e

Fig 1f

FIG. 1 : Estimation of the impulse response of a dipole.

Table 1: The average and range of error in the estimated impulse response over ten runs for the three signal to noise ratios as indicated in column 1.

		ERROR						
		Constraint on residual		CL minimization			Bayesian approach	
		I Weight. matrix	C Weight. matrix	I Weight. matrix	C Weight. matrix	I & C Weight. matrs.	I Weight. matrix	C Weight. matrix
33 dbs	Ave.	.07	.19	.22	.48	.14	.34	.31
	Range	.05-.1	.08-.21	.05-1.4	.09-3.6	.07-.12	.24-.57	.17-.61
21 dbs	Ave.	.2	.39	.53	.25	.17	.34	.4
	Range	.09-.89	.14-1.9	.12-3.7	.16-.45	.12-.27	.27-.51	.22-.7
10 dbs	Ave.	.56	.89	.38	.32	.34	.3	.65
	Range	.2-2.9	.21-5.6	.27-.71	.23-.61	.22-.7	.23-.46	.32-1.1

Table 2: Estimating noise variance

	Estimated noise variance							
actual	reduced model method				bayesian approach			
noise	55x55		19x19		I weighting		C weighting	
var.	ave.	range	ave.	range	ave.	range	ave.	range
6.6	7.4	.58-16.	8.7	2.4-19.	2.59	1.7-3.6	3.25	2.5-4.08
1.0			1.3	.47-2.5	.54	.4-.69	.62	.45-.74
1.3			1.64	.61-3.3	1.0	.82-1.2	.91	.68-1.0

scale row 1 by 10^{-6}, row 2 by 10^{-4} and row 3 by 10^{-3}

Table 3: Number of significant values in impulse response to determine its length

order	45	50	55	60	65	70
			number of significant values			
run 0	36	42	43	45	50	44
run 1	36	42	43	46	49	44
run 2	36	42	44	46	50	45
run 3	36	42	43	45	50	44
run 4	36	42	44	45	50	44
run 5	36	42	43	45	50	44
run 6	6	11	16	14	31	11
run 7	32	38	40	41	46	38
run 8	37	42	46	47	51	47
run 9	37	42	46	49	51	47

A NOVEL METHOD FOR SOLVING EXTREME EIGENVECTORS

YU HEN HU and PIN-KUAN CHOU
Department of Electrical Engineering
Southern Methodist University
Dallas, Tx 75275

ABSTRACT

In this paper, a new method is presented for the computation of extreme eigenvectors of a positive definite, symmetric matrix. This method is based on an optimized formulation and makes use of a subspace approximation technique to facilitate iterations. When applied to adaptive signal processing problems, this method has shown significant performance improvement over other eignevector updating algorithms. This paper will focus on the derivation of the algorithm and the discussion of its properties.

I. INTRODUCTION

The task of computing extreme eigenpairs of a covariance matrix has lately received much research interests due to the potential application to high resolution signal processing techniques. For example, the Pisarenko's spectrum estimate [1] would require the computation of minimum eigenvalue and associate eigenvector of the signal covariance matrix. For the high resolution spectrum estimation, one popular model is to assume that the observed time series data is composed of sinusoids contaminated by additive noise. For such a model, the observation covariance matrix R_Y can be decomposed into two parts:

$$R_Y = R_S + R_N$$

where R_S is the signal covariance matrix and R_N is the covariance matrix of the noise process. It can be shown [2] that with the white noise assumption, $R_N=\sigma^2 I$ where σ^2 is the variance of the noise which equal to λ_1, the minimum eigenvalue of R_Y, the first p largest eigenvectors of the R_Y matrix constitute an orthonormal basis for the signal subspace, and the rest eigenvectors constitute the noise subspace. Based on the notion of signal/noise subspace decomposition, a number of subspace approximation based spectrum estimation techniques have been investigated [2]-[5]. Common to all these methods, a major comuputational task is to compute a set of eigenvectors which constitutes the orthonormal basis of the signal/noise subspace. Usually either the dimension of the signal subspace, i.e. p, or the dimension of the noise subspace, say N-p, will be much less than N. Therefore, it is desired that only the p largest, or the N-p smallest eigenvectors be computed.

This research is supported by National Science Fundation under contract ECS-8404628.

Traditionally, there are two families of eigen system solvers [6]: the similarity transform based algorithms, such as the QR, QL methods and the operator based algorithms such as the power method, inverse power methods. Since the similarity transform based algorithms always compute all the eigenpairs, when only few eigenpairs are desired, it consumes considerable redundant computations. For our purpose, the operator based eigen system solvers will be more efficient. On the other hand, for the purpose of subspace approximation, it is the subspace, not the individual eigenvectors, that is important. Consequently, orthonormal basis other than the set of corresponding eigenvectors of the subspace would be sufficient for spectral estimation purpose.

This observation motivate us, in this paper, to look into efficient methods for computing eigen subspaces. The method we proposed can be regarded as a variation of the block Rayleigh-Ritz method [6]. The key idea of this method is the introduction of a block gradient matrix which expands the dimension of the working subspace and therefore helps accelerating the convergence rate. It is shown that this method can also be formulated as a variation of block Lanczos method with reinitiation in each iteration. In scalar case, this method also coincides with a rotational method proposed by Fuhrmann & Liu [7].

The rest of this paper is organized as follows : In section II, the main algorithm is derived and its convergence properties are discussed. In section III, connection between this novel block gradient subspace iteration algorithm and other related methods will be established. Next, an application of this method to the adaptive Pisarenko's spectrum estimation will be reported and some preliminary simulation results are presented in section IV.

For convenience, we shall assumed that the matrices under consideration are positive definite. Also, all the eigenvalues are ascendingly ordered.

II. BLOCK GRADIENT SUBSPACE ITERATION

Given p mutually orthonormal vectors, our goal is to establish an iterative procedure to find the best approximation of the smallest p eigenvalues and associated eigenvectors of a positive definite matrix A. Conventionally, a Rayleigh-Ritz (RR) procedure can be applied repeatedly to accomplish this goal. The resulting procedure is known as subspace iteration (SI) method. Before we derive the proposed novel method, it is instructive to review the RR procedure and the SI algorithm.

Given an nxn matrix A and a set of p ($p \leq n$) vectors $X=[x_1,\ldots,x_p]$, the RR procedure proceeds as follows:

1. Form pxp matrix $H=X^*AX$

2. Perform eigen decomposion on H. Denote $\{(\Theta_i,b_i),i=1,..,p\}$ to be the p eigenpairs of the H matrix. The Θ_i are known as the Ritz values.

3. Compute the Ritz vectors $d_i=Xb_i$, $i=1,..,p$.

It can shown that the set of Ritz values and Ritz vectors $\{(\Theta_i,d_i),i=1,..,p\}$ is the optimal approximations to the eigenpairs of A given the subspace spanned by X. However, in practice, simple application of the RR procedure may not be provided satisfactory answer. Often, an iterative procedure is desired. To facilitate such iterations, a subspace iteration (SI) method based on the RR procedure has been developed in the past [6]. In brief, this method can be summarized below:

1. Compute $S(k)=AX(k)$

2. $S(k)=Q(k)T(k)$ where $Q(k)$ is a orthonormal set of vectors.

3. Form $\hat{H}(k)=Q^*(k)AQ(k)$

4. Compute Ritz values from $\hat{H}(k)=E(k)\mathrm{diag}[\Theta_i]E^*(k)$

5. Compute Ritz vectors $X(k+1)=Q(k)E(k)$

The Ritz values $\{\Theta_i;i=1,..,p\}$ should converge monotonically to their limits $_i$, the corresponding eigenvalues of A, as subspace iteration proceeds.

II.1 Derivation of BGSI algorithm

The method we proposed in this paper is very similar to the above SI method. However, we use an enlarged subspace for approximation in order to enhance performance. In doing so, we consider a gradient matrix defined as:

$$G(k) = AX(k) - X(k)R(k)$$

where $R(k) = X(k)^*AX(k)$. $G(k)$ is called a "gradient matrix" because in scalar case ($p=1$), this matrix reduce to a vector which has the same direction as the gradient vector corresponding to the Rayleigh quotient $r(A,\underline{x})=\underline{x}^*A\underline{x}/\underline{x}^*\underline{x}$. An important property of the $G(k)$ matrix is that

$$G^*(k)X(k) = 0$$

In other words, $G(k)$ lies in the orthogonal subspace of $X(k)$. This motivates us to consider an augmented subspace composed of the subspace spanned by both $X(k)$ as well as $G(k)$. Specifically, at each iteration, the block gradient matrix is appended to the previous estimates to form a nx2p matrix P whose columns can be used as the basis for S^{2p}.

Then the RR procedure may be applied to find the desired X(k+1), the new subspace estimate. The procedure can be summarized as follows:

Block Gradient Subspace Iteration Algorithm

Initially, X(0) is a nxp orthonormal matrix

1. $R(k) = X^*(k)\ A\ X(k)$

2. Compute $G(k) = AX(k) - X(k)R(k)$. If $||G(k)|| < \epsilon$, stop.

3. Orthonormalize $G(k) = \hat{G}(k)T(k)$ where $\hat{G}(k)$ is nxp orthonormal matrix.

4. Construct augmented subspace matrix $P(k) = [X(k)|\hat{G}(k)]$

5. Form $H(k) = P^*(k)AP(k)$

6. Eigen-decomposition of H(k): $H(k) = E(k)D(k)E^*(k)$, where $D(k) = \mathrm{diag}[\theta_i;i=1,2p]$. Take the last p Ritz values $\{\theta_i; i=1,p\}$ as the current estimate of eigenvalues. Let the associated p eigenvectors be denoted by the matrix V(k).

7. Form the Ritz vectors $X(k+1) = P(k)V(k)$, as next iteration. Let k=k+1, go to step 1.

Discussion:

1. The block gradient matrix G(k) can also be regarded as residual matrix of X(k). In other words, the norm of G(k) reflects the accuracy of the approximation to the desired eigensubspace by X. As the algorithm converges, the norm of the G(k) should approach zero. Hence, this can be used as a termination criterion. However, we also note that this may take considerable computation. A more efficient criterion is yet to be developed.

2. The orthonormalization in step 3 can be achieved by, using, say, the Gram-Schmidt procedure or QR factorization. The resulting T(k) then will be an upper triangular matrix.

3. Taking rounding error into consideration, the columns of the P(k) matrix may lose linear independnece. For this, we suggest that P(k) be reorthogonalized from time to time in step 4.

4. In step 6, we chosed the smallest p Ritz values in order to find the smallest p eigenvectors (which may correspond to a noise subspace). If the largest p eigenvectors are to be approximated, we simply take the larger p Ritz values and form V(k) from the associated eigenvectors.

II.2 Convergence of the BGSI algorithm

To show the convergence of this algorithm, we shall establish that the chosen Ritz values, $\{\Theta_i;\ i=1,p\}$ in the (k+1)-th iteration will be smaller than those of the k-th iteration. For this, let us consider the H matrix

$$H = [X\ \hat{G}]^* A [X\ \hat{G}] = \begin{vmatrix} X^*AX & X^*A\hat{G} \\ \hat{G}^*AX & \hat{G}^*A\hat{G} \end{vmatrix} = \begin{vmatrix} D & \hat{G}^*G \\ \hat{G}^*G & \hat{G}^*A\hat{G} \end{vmatrix}$$

Let the eigenvalues of the H matrix be $\{\Theta_1,..,\Theta_{2p}\}$ and the eigenvalues of the submatrix X^*AX (=D) be $\{d_1,..,d_p\}$. Since D is a submatrix of the H matrix, by repeatedly applying a classical Cauchy's interlace theorem [6], we have the following conclusion:

Theorem 1 $\Theta_i \leq d_i$ for $i = 1,2, \ldots,p$.

Due to page limitation, the proof will be omitted. Note that since the Ritz values are always greater or equal to the corresponding true eigenvalues, this suffices to prove the convergence of the algorithm.

As for the convergence rate, it is very difficult to provide an explicit estimate. Nonetheless, we want to show that given the same X(k), the BGSI will always yield a better X(k+1) than the SI method. First, we argue that the span of G(k) is the same as the span of AX(k). As a matter of fact, G(k) is the projection of AX(k) to the orthogonal complement subspace of X(k). Consequently, we have $G^*(k)X(k) = 0$. Now that the span of P(k) is the direct sum of the span of X(k) and of G(k), we conclude that the subspace spanned by AX(k) lies within the span of P(k). Intuitively, we would expect the best approximation from the subspace spanned by P(k) would be at least as good as that obtained from the subspace spanned by AX(k). Based on this argument, we may as well conclude that the convergence rate should be as good as, if not better than, the subspace approximation method. Of course, it is understood that addtional computations are necessary for the eigen decomposition of the H matrix which has twice the dimension than the SI method.

II.3 Relation to Block Lanczos algorithm

From a different point of view, the BGSI method presented above can be regarded as a variation of the block Lanczos algorithm with reinitiation in every iteration. The Lanczos method is rooted in a notion known as the Krylov subspaces which, in the block case, can be defined as:

$$S^m(X) = \text{span}\{X, AX, \ldots, A^{m-1}X\}$$

The purpose of the block Lanczos algorithm is to find the optimal approximation of the eigenpairs of the A matrix from the Krylov subspace. It turns out that if X(k) is used as an initial block vector in the block Lanczos algorithm, then our procedure will coincide with the Lanczos algorithm. However, it is true only if we reinitiate the Lanczos algorithm in every iteration. From a numerical stand point of view, it is well known that the Lanczos algorithm suffers from the problem of lossing orthogonality. The constant reintiation thus helps in alleviating such numerical drawback.

III. SCALAR GRADIENT SUBSPACE ITERATION METHOD (p=1)

The scalar gradient subspace iteration method (SGSI) is the special case of the BGSI algortihm when the block size equals to 1. The SGSI method facilitates computation of distinct extremal (maximum or minimum) eigenpair. In this section, we shall discuss a few distinct properties of the SGSI method.

In SGSI method, since p=1, we have X(k) becomes a single vector x(k) and G(k) becomes g(k). In this case, Rayleigh quotient is a scalar and g(k) is proportional to the graident vector of the Rayleigh quotient. In addtion, the matrix $H=P^{*}AP$ becomes a 2x2 symmetric matrix:

$$H = \begin{vmatrix} \underline{x}^{*}A\underline{x} & \underline{x}^{*}A\hat{g} \\ \hat{g}^{*}A\underline{x} & \hat{g}^{*}A\hat{g} \end{vmatrix}$$

where $\hat{g}$ is the normalized g(k) vector. For a 2 by 2 Hermitian matrix, its eigenvector must be of the form $[\mathrm{Cos}\ v, \mathrm{Sin}\ v]^{*}$. Thus, the new eigenvector estimate x(k+1) can be represented as a linear combination of x(k) and g:

$$x(k+1) = x(k) \cos v + \hat{g} \sin v$$

It turns out that this formula, being a special case of the BGSI method, is exactly the same as that proposed by D.Fuhrmann and B. Lin [7] previously.

In the scaler case, stronger results can be derived for convergence and convergence rate. Due to space limitation, we only give the following lemma. Detailed proofs can be found in [8].

Lemma 1. If $z^{*}(1)x(0) \neq 0$ then $z^{*}(1)x(k) \neq 0$ for all k.

Lemma 1 says that if the initial eigenvector estimate x(0) is not perpendicular to the minimal eigenvector, the following x(k) will not be perpendicular to it either.

Lemma 2. The Rayeigh Quotient sequence $r(k) = x^{*}(k)Ax(k)$ is a strictly monotonic decreasing sequence. i.e. $r(k) \leq r(k+1)$.

Lemma 3. If the minimal eigenvalue is with multiplicity one then the error angle $\psi(k) = (z(1), x(k))$ approach to zero as $k \to \infty$ at a linear convergence rate.

IV. APPLICATION TO ADAPTIVE PISARENKO SPECTRUM ESTIMATION

A major application of the BGSI method is for high resolution spectral estimation. A popular method used is the sinusoids plus noise model employed by Pisarenko [1]. In Pisarenko's method, the following steps is taken to estimate the amplitudes and frequencies from the observation covariance matrix R_y :

1. Perform eigenvalue decomposition of the N by N covariance matrix R_y. Then the number of sinusoids (p) equal to the difference between N and the multiplicity of the minimum eigenvalue.

2. Compute the eigenvector corresponding to the minimum eigenvalue λ_{min} of the R_y^{p+1} matrix and denote it by $\underline{a}$.

3. Solve equation $a(z)=1+\sum_{i=1}^{p} a_i z^{i}=0$ to find the frequencies estimates $\{w_i; i=1,..,p\}$. The amplituides then can be computed via a least square fit to the given covariance sequence.

Theoretically, with exact covariance covariance lags and ideal white noise, Pisareko's method should yield exact frequencies estimates. However, practically, it has been reported [9] that numerically Pisarenko's estimate is very sensitive to perturbation of the given covariance sequence. The degradation of these estimates will become worse if the noise process is correlated (colored noise). Recently, a number of researchers have attempted to generalize the Pisarenko's procedure such that instead of one vector, p vectors which constitute a signal or noise subspace [3] can be utilized to improve the profermance. Examples include the MUSIC method, noise subspace approximation method [4], SVD based approximation method [10], Toeplitz approximation method [2], and stochastic approximation based methods [11].

To apply BGSI to these methods, note that each of these method expect a set of eigenvectors from the covariance matrix. Then various procedures can be taken to obtain the desired frequency estimate. Thus, the BGSI can be regarded as a computation frontend to these methods. For example, given X(k), an enhanced MUSIC type spectrum can be evaluated as [4]

$$P_{music} = 1/|V^*(w)X(k)|^2$$

Another possible application is for the adaptive Pisarenko procedure where the covariance matrix is constantly being updated by incoming observations. For this, we have experimented with the SGSI method and observed significant performance improvement over other methods. As the details shall appear in a later publication, two simulation examples will be included here as dipicted in figure 1 and figure 2. In these examples, two other adaptive algorithms are compared with the SGSI method in either single or double harmonic retreival cases.

Example 1

we consider a single sinusoid of frequency 0.25π (45^{o}) at a signal-to-noise ratio (SNR) of 20 dB. The SGSI method is compared with the unin-norm algorithm presented by Thompson [12]. The result is dipicted in Fig 1. we observed that SGSI converges much faster to the true angle.

Example 2

Two equal power sinusoids of frequencies 0.167π (30^{o}) and 0.333π(60^{o}) at 20 dB SNR are considered in this example. The plots of Fig.2 show that in steady-state, the SGSI is more stable comparing with least square type algorithm which reported by Reddy, Egardt and Kailath [13].

V. CONCLUSION

In this paper,we have proposed a novel method for finding the extreme eigenvectors. With the notion of signal/noise subspace decomposition, the proposed method can be applied to resolution spectrum estimation problem and has shown significant improvement over other eigenvector updating algorithm.

REFERENCE

[1] V.F. Pisarenko, "The retrieval of harmonics from a covariance function," Geophysical J. Royal Astronomical Soc., Vol. 33, pp. 347-366, 1973.

[2] S.Y. Kung, "A Toplitz Approximation Method and Some Applications," Int'l Symp. on Math.Theory of networks and Syst., Santa Monica, CA August 5-7, 1981.

[3] S.Y. Kung and Y.H. Hu, "Improved Pisareko's Sinusoidal Spectrum estimate via SVD Subspace Approximation Method," Proc. 21-th CDC, Oriando, Florida, Dec. 8-10 1982.

[4] R. Schmidt, "Multiple emitter locations and signal parameter estimateion," Proc. RADC spectral estimation workshop, Rome, NY, 1979, pp. 243-258.

[5] G. Bienvenu and L. Kopp, "Adaptive High Resolution Spatial Discrimination of Passive Sources," in "Underwater Acoustics and Signal Processing", Ed. L.Bjorno, D. Reidel, 1981.

[6] B.N. Parlett, The symmetric eigenvalue problem, Englewood Cliffs, NJ: Prentice-Hall, 1980.

[7] D. Fuhrmann and B. Lin, "An iterative Algorithm for Locating the Minimal Eigenvector of a Symmtric Matrix", Proc. ICASSP 84,p.45.8.1, San Deigo, Ca. March 1984.

[8] Y.H. Hu and P.K. Chou, " A novel method for finding extreme eigenvectors," Southern Methodist University, EE Tech. Rept., Oct., 1985.

[9] T.J. Ulrych and R.W. Clayton, "Time series modelling and maximum entropy," Phys.Earth Plantetary Interiors vol.12, pp. 188-200, Aug. 1976.

[10] D.W. Tufts and R. Kumaresan, "Estimation of frequencies of multiple sinusoids," Proc. IEEE, vol. 70, no.9, Sep. 1980, pp. 975-989.

[11] S.Y. Kung and K.S. Arun, "Approximate realization methods for ARMA spectral estimation," IEEE Symp. Circuits Syst., Newport Beach, CA. May 1983.

[12] P. Thompson, "An adaptive spectral analysis technique for unbiased frequency estimation in the presence of white noise," Presented at the 13-th Asilomar conf. circuit, syst., comput., Nov. 1979.

[13] V.U. Reddy, B. Egardt, T. Kailath, "Least squares type algorithm for adaptive implementation of Pisarenko's harmonic retrieval method," IEEE Trans. ASSP, vol. 30, No. 3, June 1982, pp. 399-405.

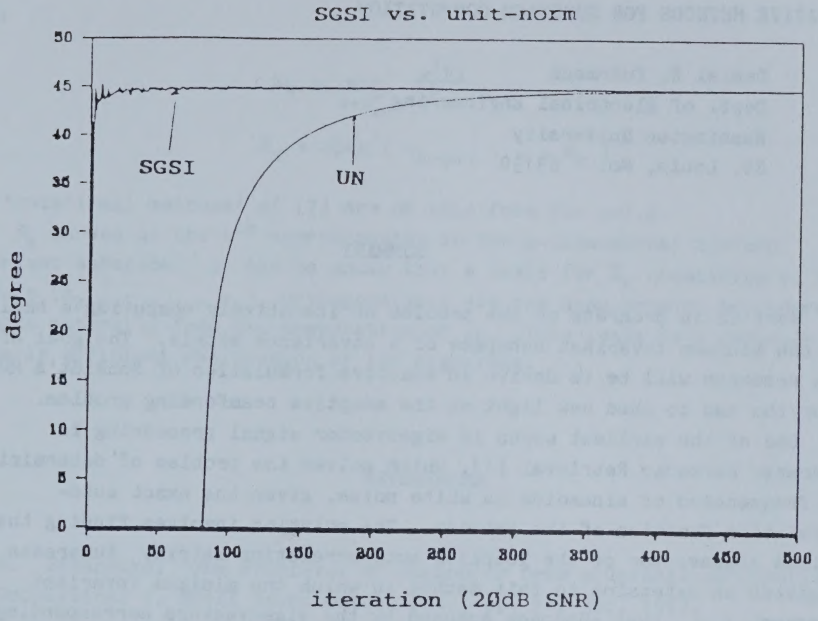

SGSI vs. unit-norm

Fig 1

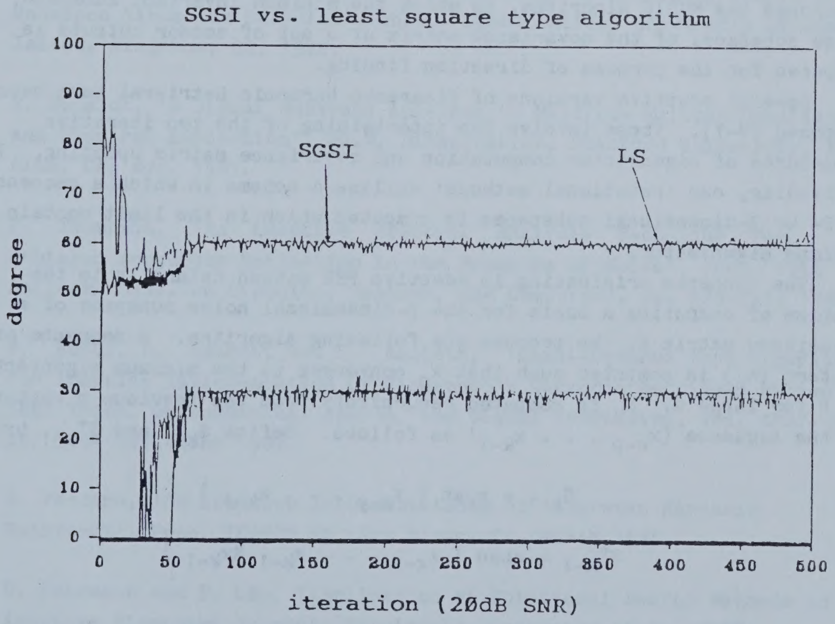

SGSI vs. least square type algorithm

Fig 2

ITERATIVE METHODS FOR SUBSPACE COMPUTATION

Daniel R. Fuhrmann
Dept. of Electrical Engineering
Washington University
St. Louis, Mo. 63130

SUMMARY

Work is in progress on the problem of iteratively computing a basis for the minimum invariant subspace of a covariance matrix. The goal of this research will be to derive an adaptive formulation of Schmidt's MUSIC algorithm and to shed new light on the adaptive beamforming problem.

One of the earliest works in eigenvector signal processing is Pisarenko Harmonic Retrieval [1], which solved the problem of determining the frequencies of sinusoids in white noise, given the exact autocorrelation function of the process. The solution involves finding the minimum eigenvector of the Toeplitz autocorrelation matrix. Kumaresan [2] suggested an extension to this method in which the minimum invariant subspace, i.e., that subspace spanned by the eigenvectors corresponding to the smallest eigenvalues, is computed. Independently, Schmidt [3] developed the MUSIC algorithm, in which the minimum invariant subspace, or noise subspace, of the covariance matrix of a set of sensor outputs is computed for the purpose of direction finding.

Several adaptive versions of Pisarenko Harmonic Retrieval have been proposed [4-7]. These involve the intertwining of the two iterative procedures of eigenvector computation and covariance matrix updating. In particular, our 'rotational methods' utilize a scheme in which a sequence of 2- or 3-dimensional subspaces is computed which in the limit contain the minimum eigenvector.

The concepts originating in adaptive PHR extend naturally to the problem of computing a basis for the p-dimensional noise subspace of a covariance matrix R. We propose the following algorithm. A sequence of vectors $\{x_n\}$ is computed such that x_n converges to the minimum eigenvector of R for large n. x_k is computed recursively from the previous p vectors in the sequence $(x_{k-p} \ldots x_{k-1})$ as follows. Define S_{k-1} and S^{+}_{k-1} by

$$S_{k-1} = \text{span} \{ x_{k-p} \ldots x_{k-1} \} \tag{1}$$

$$S^{+}_{k-1} = \text{span} \{ x_{k-p} \ldots x_{k-1}\ Rx_{k-1} \} \tag{2}$$

Then

$$x_k = \min_{x \in S_{k-1}^{+}} \frac{x^T R x}{x^T x} \qquad (3)$$

and

$$S_k = \text{span}\ \{\ x_{k-p+1}\ \cdot\ \cdot\ \cdot\ x_k\ \} \qquad (4)$$

The 'rotational methods' of [7] are of this form for p=1,2.

S_k serves as the k^{th} approximation to the p-dimensional minimum invariant subspace. It can be shown that a basis for S_k consisting of the vectors $(x_{k-p+1} \cdot \cdot \cdot x_k)$, orthogonalized via the Gram-Schmidt procedure, results naturally from the computation of x_k. This leads to a computationally efficient realization of the algorithm.

REFERENCES

1. V. Pisarenko, 'The Retrieval of Harmonics from a Covariance Function', Geophysical J. Royal Astronomical Soc., pp. 347-366, 1973.

2. R. Kumaresan, 'Estimating the Parameters of Exponentially Damped or Undamped Sinusoids in Noise', Ph.D. Dissertation, University of Rhode Island, Kingston, RI, 1982.

3. R. Schmidt, 'A Signal Subspace Approach to Multiple Emitter Location and Spectral Estimation', Ph.D. Dissertation, Stanford University, Palo Alto, CA, Nov. 1981.

4. P. Thompson, 'An Adaptive Spectral Analysis Technique for Unbiased Frequency Estimation in the Presence of Noise', Proc. 13th Asilomar Conf. on Circuits, Systems, and Computers, pp. 529-533, 1980.

5. V. Reddy, B. Egardt, and T. Kailath, 'Least-Squares Type Algorithm for Adaptive Implementation of Pisarenko's Harmonic Retrieval Method', IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. ASSP-30, pp. 399- 405, June 1982.

6. R. Vaccaro, 'On Adaptive Implementations of Pisarenko Harmonic Retrieval', Proc. ICASSP 84, San Diego, CA, March 1984.

7. D. Fuhrmann and B. Liu, 'Application of Rotational Search Methods to Adaptive Pisarenko Harmonic Retrieval', presented at the IEEE International Symposium on Information Theory, Brighton, England, June 1985.

A FAST PRONY ALGORITHM

J.R. CRUZ AND R.J. MULHOLLAND
School of Electrical Engineering and Computer Science
The University of Oklahoma
Norman, OK 73019

ABSTRACT

Prony's method for fitting exponentials to equally spaced data samples is a suboptimum technique for determining the values of parameters of a linear combination of exponential functions. The technique reduces the difficult nonlinear least squares problem to the solution of two sets of linear equations and the rooting of a polynomial. Although elegant, this technique provides no criterion for determing the number of exponentials required for the fitting. Model order determination can be done either by repeated use of the method or by using eigenvector or singular value decompositions, or information theoretic criteria. For all of these approaches the computational load is heavy, particularly for large number of data samples. We present a fast recursive Prony algorithm which estimates the parameters of a linear combination of exponential functions including its order. It is shown that the Berlekamp-Massey algorithm for linear shift register synthesis provides a general solution to the problem of selecting the exponential model of smallest order.

INTRODUCTION

Almost two hundred years ago, Gaspard Riche, Baron de Prony [1] proposed a method for exponential fitting of equally spaced data samples. This elegant technique still attracts today a great deal of attention [2]-[12]. There are two main reasons for such interest. First the simplicity of Prony's method makes it attractive as a suboptimum solution to a difficult nonlinear least squares problem. Secondly, the problem is of great interest in communications, seismology, underwater acoustics and spectral estimation. The method, however, suffers from at least one serious difficulty. It provides no criterion for determining the number of exponentials required for the fitting. Therefore, before Prony's algorithm is applied to a data set, prior knowledge of the number of modes present in the data is required. Several techniques have been proposed to handle this problem: the Householder orthogonalization method [7], the eigenvalue method [8] and more recently the singular value decomposition method [10]. Such decompositions are known to present some numerical difficulties for matrices larger than 200 and the computational complexity of such algorithms is on the order of N^3 [13].

In this paper we propose a fast recursive Prony algorithm which provides model order determination. The algorithm is recursive, can handle large sets of data and its computational complexity is on the order of N^2. Further reductions in complexity could be obtained by using divide and conquer techniques as suggested by Blahut [14]. It is shown that the problem of estimating the number of modes and associated eigenvalues present in a data set amounts to the synthesis of the shortest linear feedback shift register capable of generating such data set. Once the problem is interpreted in this manner then the Berlekamp-Massey algorithm [15] for shift register synthesis can be used. This algorithm which is widely used in the area of error control coding, and therefore, usually associated with finite fields, has not attracted much attention in the real or complex fields.

STATEMENT OF THE PROBLEM

Consider the uniform spaced samples $x(kT) = x(k)$ of a stationary sum of exponentials

$$x(t) = p_1 \exp(\lambda_1 t)+\ldots+p_n \exp(\lambda_n t) \qquad (1)$$

where n is a fixed integer to be determined. The sampled version of (1) can be written as

$$x(k) = p_1\mu_1^k +\ldots+p_n\mu_n^k \qquad (2)$$

where k is any integer and

$$\mu_i = \exp(\lambda_i T) \qquad i = 1,\ldots,n \qquad (3)$$

Writing (2) for $k = 0, 1, \ldots, (m-1)$ gives

$$\begin{aligned} x(0) &= p_1 +\ldots+p_n \\ x(1) &= p_1\mu_1 +\ldots+p_n\mu_n \\ x(2) &= p_1\mu_1^2 +\ldots+p_n\mu_n^2 \\ &\;\;\vdots \\ x(m-1) &= p_1\mu_1^{m-1} +\ldots+p_n\mu_n^{m-1} \end{aligned} \qquad (4)$$

which is a system with 2n unknown constants. Prony's method is an algorithm for solving (4) which assumes a priori knowledge of the value of n.

In this paper we present a recursive fast algorithm for computing coefficients p_i, the eigenvalues μ_i and the number of modes n.

A NEW INTERPRETATION OF PRONY'S METHOD

Consider the alternative representation of (4) given by the vector matrix equation

$$\underline{x} = \underline{V}_m \, \underline{p} \qquad (5)$$

where the vector $\underline{x}$ is

$$\underline{x} = [x(0)\ x(1)\ \ldots\ x(m-1)]^T \qquad (6)$$

the vector $\underline{p}$ is

$$\underline{p} = [p_1\ p_2\ \ldots\ p_n]^T \qquad (7)$$

and the matrix $\underline{V}_m$ is

$$\underline{V}_m = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \mu_1 & \mu_2 & \dots & \mu_n \\ \mu_1^2 & \mu_2^2 & \dots & \mu_n^2 \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \mu_1^{m-1} & \mu_2^{m-1} & & \mu_n^{m-1} \end{bmatrix} \tag{8}$$

Let the ℓ-vector $\underline{x}^1$ as in [11] denote the first ℓ data points. Then the first ℓ equations in (4) can be written

$$\underline{x}^1 = \underline{V}_\ell \, \underline{p} \tag{9}$$

Similarly

$$\underline{x}^2 = \underline{V}_\ell \, \underline{U} \, \underline{p} \tag{10}$$

where $\underline{x}^2$ denotes the second through the $\ell + 1$ data points and $\underline{U}$ is the diagonal matrix of the eigenvalues

$$\underline{U} = \text{diag} \{\mu_1, \mu_2, \dots, \mu_n\} \tag{11}$$

In general the i-th view of the data is given by

$$\underline{x}^i = \underline{V}_\ell \, \underline{U}^{i-1} \, \underline{p} \quad i = 1, 2, \dots., n+1 \tag{12}$$

where $\underline{U}^i$ is the i-th power of the matrix $\underline{U}$. Multiplying $\underline{x}^1$ by c_n, $\underline{x}^2$ by c_{n-1}, and so on, until $\underline{x}^n$ is multiplied by c_1 and $\underline{x}^{n+1}$ by 1, and adding, gives

$$c_n\underline{x}^1 + c_{n-1}\underline{x}^2 + \dots + c_1\underline{x}^n + \underline{x}^{n+1} = \underline{V}_\ell(c_n\underline{I} + c_{n-1}\underline{U} + \dots + c_1\underline{U}^{n-1} + \underline{U}^n)\underline{p} \tag{13}$$

The key step in Prony's algorithm consists in realizing that the right hand side of (13) can be set equal to zero provided that

$$c_n\underline{I} + c_{n-1}\,\underline{U} + \dots + c_1\underline{U}^{n-1} + \underline{U}^n = \underline{0} \tag{14}$$

Multiplying both sides of (14) by $\underline{U}^{-n}$ gives

$$\underline{I} + c_1\underline{U}^{-1} + \dots + c_n\underline{U}^{-n} = \underline{0} \tag{15}$$

which shows that the polynomial

$$C(D) = c_0 + c_1 D + c_2 D^2 + \dots + c_n D^n \tag{16}$$

where $c_0 = 1$, must have for its roots μ_i^{-1}, for $i = 1, \ldots, n$. Such a polynomial can be written in its factored form as

$$C(D) = \prod_{i=1}^{n} (1-D\mu_i) \tag{17}$$

and (13) can be written as

$$x(i) = -\sum_{j=1}^{n} c_j\, x(i-j) \quad i=n,\ n+1,\ \ldots,\ \ell+n-1 \tag{18}$$

In general $\ell > n$ $(m>2n)$ and the system (18) is overdetermined. One popular version of the algorithm due to Whittaker and Robinson [16], finds a least squares estimate for the coefficients c_i, for $i = 1, \ldots, n$, assuming of course, prior knowledge of n. The algorithm presented here, does not assume that n is known, and recognizes that (18) is the equation of an autoregressive filter which may be implemented as a linear feedback shift register. Viewed this way the problem becomes one of designing the smallest linear feedback shift register shown in Fig. 1 that generates the known data sequence. The Berlekamp-Massey algorithm [15] provides a design procedure to obtain the shift register length L and the feedback connection polynomial C(D). The algorithm is valid for data samples and coefficients being in the same field, which can be finite or infinite such as the real or complex number field.

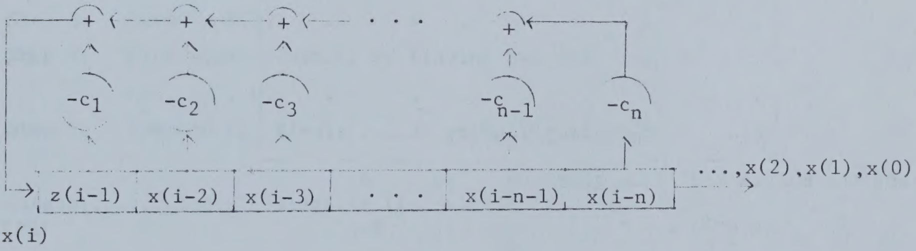


Figure 1. Linear feedback shift register implementation of Eq. 18.

The problem is now reduced to finding coefficients $p_1, p_2, \ldots, p_n$ of the linear combination in (2). Following Massey [15], we first observe that

$$\frac{1}{1-\mu_i D} = 1 + \mu_i D + \mu_i^2 D^2 + \ldots \tag{19}$$

and multiplying both sides by p_i and summing gives

$$\sum_{i=1}^{n} \frac{p_i}{1-\mu_i D} = x(0) + x(1)\,D + x(2)\,D^2 + \ldots \tag{20}$$

where the right hand side is the polynomial X(D). Multiplying both sides by C(D) gives

$$\sum_{i=1}^{n} p_i \prod_{\substack{j=1\\ j\neq i}}^{n} (1-\mu_j D) = X(D)C(D) \tag{21}$$

and letting

$$R(D) = \sum_{i=1}^{n} p_i \prod_{\substack{j=1\\ j\neq i}}^{n} (1-\mu_j D) = r_0 + r_1 D + \ldots + r_{n-1} D^{n-1} \tag{22}$$

equation (21) can be written as

$$R(D) = X(D)\,C(D) \tag{23}$$

The coefficients of R(D) can be obtained from the matrix equation

$$\begin{bmatrix} r_0 \\ r_1 \\ \cdot \\ \cdot \\ \cdot \\ r_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \ldots & 0 & 0 \\ c_1 & 1 & \ldots & 0 & 0 \\ \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot \\ \cdot & \cdot & & \cdot & \cdot \\ c_{n-1} & c_{n-2} & \ldots & c_1 & 1 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \cdot \\ \cdot \\ \cdot \\ x(n-1) \end{bmatrix} \tag{24}$$

and the matrix $R(\underline{U}^{-1})$ is given by

$$R(\underline{U}^{-1}) = \underline{P}\ \underline{T} \tag{25}$$

where

$$\underline{P} = \mathrm{diag}\,\{p_1, p_2, \ldots, p_n\} \tag{26}$$

and

$$\underline{T} = \mathrm{diag}\{\prod_{j=2}^{n} (1-\frac{\mu_j}{\mu_1}), \prod_{\substack{j=1\\ j\neq 2}}^{n} (1-\frac{\mu_j}{\mu_2}), \ldots, \prod_{j=1}^{n-1} (1-\frac{\mu_j}{\mu_n})\} \tag{27}$$

The coefficients $p_1, p_2, \ldots, p_n$ can be computed from

$$\underline{P} = R\,(\underline{U}^{-1})\ \underline{T}^{-1} \tag{28}$$

The extended Prony's method reduces a nonlinear least squares problem to the solution of two linear least squares problems and the rooting of a polynomial. The number of computations necessary to execute the algorithm is proportional to m^3. The technique outlined in this section explores the Hankel structure of the data matrix so that no matrix inversion is necessary. The number of computations necessary to execute this algorithm is proportional to m^2.

A FAST PRONY ALGORITHM

The extended Prony's method is a three step algorithm which requires the computation of pseudo-inverses of nxm and nx(m-n) matrices and the rooting of a n-th order polynomial. The computational load associated with the pseudo-inversion of two matrices may be excessive, particularly for large values of m. In the algorithm presented here, both pseudo-inverses are avoided. The first generalized inversion is required in the computation of the polynomial coefficients; it can be avoided by using the Berlekamp-Massey algorithm. The second generalized inversion is required for the computation of the coefficients of the linear combination of exponentials; this too can be avoided by using the equations derived in the previous section.

A flow diagram of the computations of the fast Prony algorithm developed here is shown in fig. 2; a step-by-step description of the algorithm follows:

Step 1: Input m data points

Step 2: Find C(D) using the Berlekamp-Massey algorithm.

Step 3: Compute R(D).

Step 4: Find eigenvalues μ_k by finding the zeros of C(D), $k=1, \ldots, n$.

Step 5: Compute p_k, $k = 1, \ldots, n$ using the equation

$$\mathrm{diag}\{p_1,\ldots,p_n\}=R(\underline{U}^{-1})\,\mathrm{diag}\{1/\prod_{j=2}^{n}\left(1-\frac{\mu_j}{\mu_1}\right),1/\prod_{\substack{j=1\\ j\neq 2}}^{n}\left(1-\frac{\mu_j}{\mu_2}\right),\ldots,1/\prod_{j=1}^{n-1}\left(1-\frac{\mu_j}{\mu_n}\right)\}$$

NUMERICAL EXAMPLE

In order to illustrate the behavior of the fast Prony algorithm discussed above, we applied it to the seismic source wavelet data used by Kung [17]. The original sequence and the impulse response obtained by the fast Prony algorithm with the convergence parameter d set to 0.01 in the Berlekamp-Massey algorithm are given in Table 1. The impulse response data as well as the approximation are plotted in fig. 3. It is easily seen that the proposed algorithm does closely approximate the original wavelet. Since the order of the approximation selected by the Berlekamp-Massey algorithm was n=9, the first nine Markov parameters are identical to the original sequence. In order to further visualize the performance of the algorithm, the absolute error has been plotted in fig. 4.

In addition, the squared error of the impulse response, i.e.

$$\sum_{k=0}^{m-1} \left| h(k) - \sum_{i=1}^{n} p_i \mu_i^k \right|^2 \qquad (29)$$

was computed to be 2.2970×10^{-4}

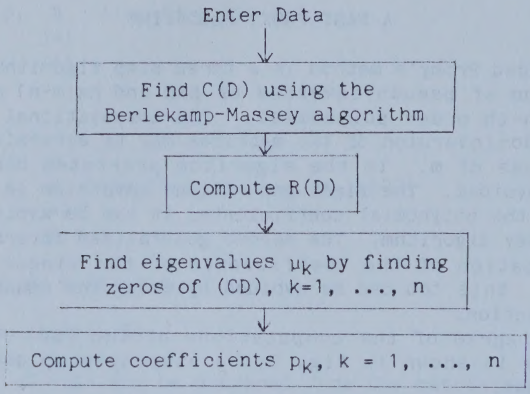

Enter Data
Find C(D) using the Berlekamp-Massey algorithm
Compute R(D)
Find eigenvalues μ_k by finding zeros of (CD), k=1, ..., n
Compute coefficients p_k, k = 1, ..., n

Figure 2. Fast Prony Algorithm

CONCLUSIONS

In this paper we presented a fast algorithm for computing the parameters of a linear combination of exponential functions including its order. A connection between the old Prony's method and the Berlekamp-Massey algorithm was established, which led to the development of a fast recursive algorithm. The main features of this algorithm are: i) it provides model order determination; ii) its complexity is on the order of m^2.

ACKNOWLEDGEMENTS

This work was supported in part by a Summer Research Fellowship from the University of Oklahoma. The authors wish to thank Mr. Mansour Tahernezhadi for generating the figures in this paper.

REFERENCES

[1] G.R. B. Prony, "Essai experimental et analytique, etc.," Paris, **J. de L'Ecole Polytechnique**, vol. 1, cahier 2, pp. 24-76, 1795.

[2] A.J. Poggio et al., "Evaluation of a processing technique for transient data," **IEEE Trans. Antennas and Propogation**, vol. AP-26, no. 1, pp. 165-173, January 1978.

[3] S.M. Kay and S.L. Marple, "Spectrum analysis-a modern perspective," **Proc. IEEE**, vol. 69, no. 11, pp. 1380-1419, November 1981.

[4] F.B. Hildebrand, **Introduction to Numerical Analysis**, New York: McGraw-Hill, 1956.

[5] R.W. Hamming, **Numerical Methods for Scientists and Engineers**, 2nd Edition, New York: McGraw-Hill, 1973.

[6] W. Simon, **Mathematical Techniques for Physiology and Medicine**, New York: Academic Press, 1972.

[7] A.S. Householder, "On Prony's method of fitting exponential decay curves," **O.R.N.L. Technical Report**, ORNL-455, Oak Ridge, Tennessee: Oak Ridge National Laboratory, 1949.

[8] M.L. Van Blaricum and R. Mittra, "Problems and solutions associated with Prony's method for processing transient data," **IEEE Trans. Antennas and Propagation**, vol. AP-26, no. 1, pp. 174-182, January 1978.

[9] R. Kumaresan and D.W. Tufts, "Improved spectral resolution III: efficient realization," **Proc. IEEE**, vol. 68, no. 10, pp. 1354-1355, October 1980.

[10] T.L. Henderson, "Geometric methods for determing system poles from transient response," **IEEE Trans. Acoustics, Speech, and Signal Processing**, vol. ASSP-29, no. 5, pp. 982-988, October 1981.

[11] J.L. Crittenden et al., "Sampling and data analysis properties of Prony's method of applied to model identification," **Int. J. Systems Science**, vol. 14, no. 5, pp. 571-584, May 1983.

[12] R. Kumaresan et al., "A Prony method for noisy data: choosing the signal components and selecting the order in exponential signal models," **Proc. IEEE**, vol. 72, no. 2, pp. 230-233, February 1984.

[13] V.C. Klema and A.J. Laub, "The singular value decomposition: its computation and some applications," **IEEE Trans. Automatic Control**, vol. AC-25, no. 2, pp. 164-176, April 1980.

[14] R.E. Blahut, **Theory and Practice of Error-Control Codes**, Reading, Mass.: Addison-Wesley, 1983.

[15] J.L. Massey, "Shift-register synthesis and BCH decoding," **IEEE Trans. Information Theory**, Vol. IT-15, no. 1, pp. 122-127, January 1969.

[16] E.T. Whittaker and G. Robinson, **The Calculus of Observations**, London: Blackie and Son, 1924.

[17] S. Kung, "A new identification and model reduction algorithm via singular value decompositions," in **Proc. 12th Asilomar Conf. Circuits, Systems and Computers**, Pacific Grove, CA, pp.705-714, 1978.

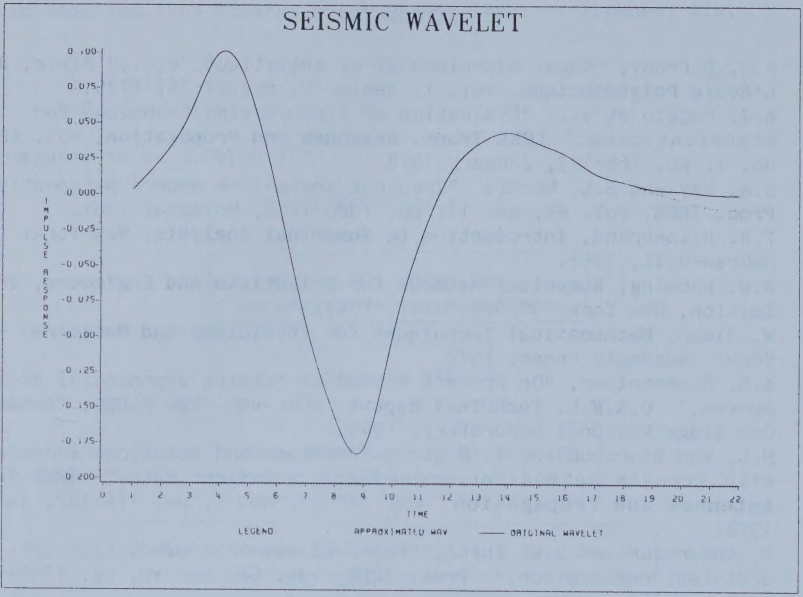


Figure 3. Seismic wavelet and its approximation

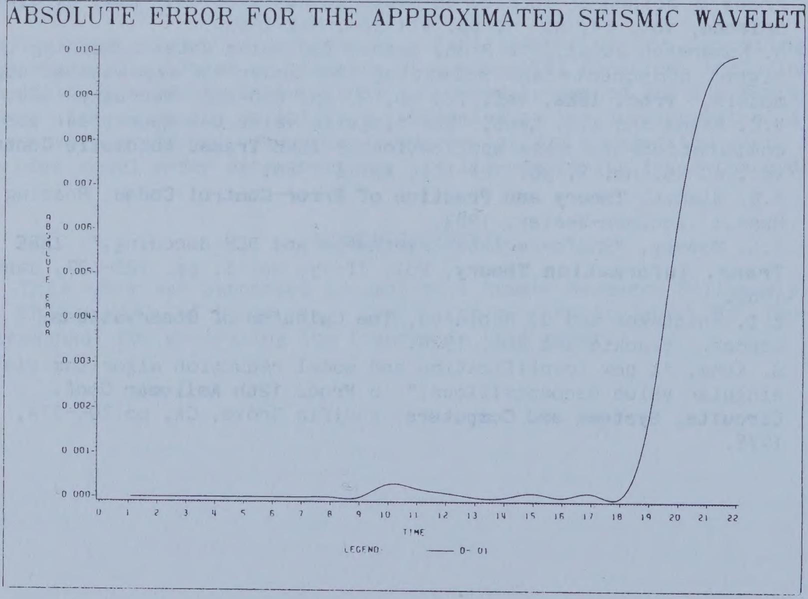


Figure 4. Absolute error for the approximated seismic wavelet

TABLE I: A Seismic Wavelet and its Fast Prony Approximation

	Original Sequence	Fast Prony Algorithm d = 0.01
k	h(k)	h(k)
1	0.0030	0.0030
2	0.0250	0.0250
3	0.0576	0.0576
4	0.1000	0.1000
5	0.0758	0.0758
6	0.	0.0000
7	-0.0909	-0.0909
8	-0.1563	-0.1563
9	-0.1789	-0.1789
10	-0.1000	-0.1003
11	-0.0303	-0.0305
12	0.0156	0.0155
13	0.0364	0.0364
14	0.0438	0.0438
15	0.0364	0.0365
16	0.0281	0.0281
17	0.0152	0.0153
18	0.0094	0.0094
19	0.0045	0.0027
20	0.0031	-0.0033
21	0.0010	-0.0083
22	0.0006	-0.0093

A NEW IMPLEMENTATION OF THE CONJUGATE GRADIENT METHOD FOR NONLINEAR SYSTEMS

BARRY J. SULLIVAN and JORGE M. NOCEDAL*
Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, Illinois 60201

ABSTRACT

The conjugate gradient method provides an iterative technique for function minimization. At each iteration, a new search direction is calculated based on the current gradient of the function and the previous search direction. Several formulas for the scalar weighting factor β applied to the previous search direction have been developed over the years, all of which make use of the gradient of the function evaluated at the current solution and the previous one. A generalization that uses the gradient evaluated at any point along the search direction is investigated here. For quadratic function minimization, this new formula for β produces a set of conjugate search directions, as required.

SUMMARY

The conjugate gradient method was originally developed to solve linear systems by minimizing a quadratic function. When the method is generalized to minimize non-quadratic functions, it can be applied to solving certain nonlinear systems as well. However, implementations of the method that are essentially equivalent for solving linear systems exhibit quite different behavior in the more general case. A new implementation capable of ensuring certain desirable properties is presented here.

The conjugate gradient method provides an iterative technique for function minimization. At each iteration, a new search direction d_{k+1} is calculated as

$$d_{k+1} = -g_{k+1} + \beta_{k+1} d_k,$$

where g_{k+1} is the gradient of the function evaluated at the solution of the $(k+1)$-th step. Several formulations for the scalar weighting factor β have been developed over the years, based on quadratic function minimization [1]-[3]. All of these formulations–Hestenes-Stiefel, Polak-Ribiere, and Fletcher-Reeves–make use of the gradient of the function evaluated at the current solution and the previous one. A generalization that uses the gradient evaluated at any point along the search direction is investigated here. The calculation of β is carried out as

$$\beta_{k+1} = \frac{g_{k+1}^T (g_{k+1} - g_*)}{d_k^T (g_{k+1} - g_*)},$$

where g_* is the gradient of the function at any point along the search direction d_k. This reduces to the Hestenes-Stiefel formulation when $g_* = g_k$. For quadratic function minimization, this new formula for β produces a set of conjugate search directions, as required.

---

* The work of this author was supported in part by the National Science Foundation Grant No. DCR-8401903.

Differences in the choice of β are evident as the convergence of the iteration is studied, both in terms of rates of convergence and failures to converge for some functions. The new formula for β allows flexibility in applying constraints on β to improve convergence, which is accomplished through the choice of points at which g_* is evaluated. For example, Powell has shown that it may be desirable to constrain β to be non-negative to assure convergence [4]. The proposed method allows such a constraint to be implemented without simply setting $\beta = 0$ and taking a pure gradient step whenever the calculation of β yields a negative value. Instead, an alternate value for g_* can be used, resulting in a positive value of β. Other constraints can be similarly implemented with candidates for g_* drawn from the intermediate points of the line search used to determine the current solution. Trials using the Moré set of test functions [5] indicate that the new approach compares favorably with other methods for calculating β.

REFERENCES

[1] Hestenes, M. R., and Stiefel, E. L., "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Standards,* Section B, no. 49, pp. 409-436, 1952.

[2] Polak, E., and Ribiere, G., "Note sur la convergence de methodes de directions conjugées," *Rev. Francaise Informat Recherche Operationelle,* 3e Année, no. 16, pp. 35-43, 1969.

[3] Fletcher, R., and Reeves, C. M., "Function minimization by conjugate gradients," *Computer J.,* vol. 7, pp. 149-154, 1964.

[4] Powell, M. J. D., "Nonconvex minimization calculations and the conjugate gradient method," *1983 Dundee Biennial Conf. on Num. Anal.*

[5] Moré, J. J., Garbow, B. S., and Hillstrom, K. E., "Testing unconstrained optimization software," *ACM Trans. Software,* vol. 7, pp. 17-41, 1981.

CONTROLLABILITY OF SYSTEMS WITH LARGE DELAYS

A. S. C. SINHA
Purdue University at Indianapolis
P. O. Box, 647, Indianapolis, Indiana 46223

ABSTRACT

In this paper, using the comparison method, we discuss the controllability of functional differential equations with delays in control and large time-lags in states.

INTRODUCTION

A considerable amount of literature has been devoted to the problems of controllability of systems with small delays in states and controls, see for example [1], - [5]. This paper develops criteria for controllability of systems with large time delays which can be easily verified. We consider the nonlinear delay systems

$$\dot{x}(t) = L(t,x_t) + B(t)u_t + f(t,x_t,u_t) \tag{1}$$

where $L(t,\phi)$ is continuous in t, linear in ϕ and is given by

$$L(t,\phi) = \sum_{k=o}^{N} A_k(t)\ \phi(-\tau_k(t)) + \int_{-\infty}^{o} \eta(s)\ \phi(s)ds \tag{2}$$

$$x(t) = \phi(t),\ t \text{ in } [-\gamma,0] \tag{3}$$

$$u(t) = v(t),\ t \text{ in } [-h,0] \tag{4}$$

and where each $A_k(t)$, $B(t)$ and $\eta(t)$ is a continuous matrix function for $-\infty < t < \infty$, $0 \leq \tau_k(t) < \gamma$, for all k, $h < \gamma$. The symbol $f(t,x_t,u_t)$ denotes the function

$$f\colon I \times W_2^{(1)}\ ([-\gamma,0],\ B^n) \times L_2^{loc}\ ([-h,\infty),\ E^m) \to E^n,$$

where $W_2^{(1)}$ is the Sobolev space of absolutely continuous functions whose derivatives are square integrable and L_2^{loc} $([-h,\infty),\ E^m)$ denotes the space of locally square integrable functions $u\colon [-h,\infty) \to E^m$. Let $B^n = B^n([-\gamma,0],\ E^n)$ be the Banach space of functions which are continuous on $[-\gamma,0]$ and such that

$$||\phi||_\gamma \overset{def}{=} \sup_{\theta\ \varepsilon\ [-\gamma,o]} |\phi(\theta)| + \int_{-\gamma}^{o} q(\theta)|\phi(\theta)|d\theta < \infty \tag{5}$$

Let E^n be n-dimensional linear vector space with norm $|\cdot|$, $q\colon [-\gamma,0] \to (0,\infty)$ be Lebesque integrable on $[-\gamma,0]$, positive and nondecreasing on $[-\gamma,0]$. For any $t\ \varepsilon\ I$ and any $x\colon [t-\gamma,t] \to E^n$ let $x_t\colon [-\gamma,0] \to E^n$ be defined by $x_t(\theta) = x\ (t+\theta)$, $\theta\ \varepsilon\ [-\gamma,0]$.

CONTROLLABILITY RESULTS

Consider a linear part of control system (1)

$$\dot{x}(t) = L(t,x_t) + B(t)\ u_t;\ t\ \varepsilon\ I$$
$$x(t) = \phi(t) \text{ in } [-\gamma,0]. \tag{6}$$

Let $\tau_1(t), \tau_2(t), \ldots, \tau_N(t)$ be nonnegative continuous functions of t. Suppose there is function $\nu(t)$, positive and continuously differentiable for $t \geq t_o$ satisfying

$$\tau_i(t) \leq \nu(t),\ t \geq t_o,\ i = 1,2,\ldots,N \qquad (7)$$

$$0 \leq \frac{d\nu(t)}{dt} \leq a < 1,\ t \geq t_o \qquad (8)$$

and, in addition let

$$h \leq \nu(t) \text{ for all } t \geq t_o \qquad (9)$$

The controls u are square integrable with values in the unit cube V^m:

$$V^m = \{u : u \in E^m,\ |u_j| \leq 1,\ j = 1,\ldots,m\} \qquad (10)$$

$X(t,s)$ be the fundamental matrix of $\dot{x}(t) = A_o(t)x(t)$, such that

$$||X(t,s)|| \leq C\, e^{-\lambda(t-s)},\ t_o \leq s \leq t < \infty \qquad (11)$$

Theorem. Suppose that conditions (7) - (11) hold. In addition, if

(i) B(t) is continuously differentiable

(ii) $W(t_1) \overset{\text{def}}{=} \int_{t_o}^{t_1} \Phi(s)\, \Phi^T(s)\,ds$ is nonsingular

$$B^*(t) \overset{\text{def}}{=} \begin{cases} B(t) & \forall\ t \in [t_o, t_1] \\ 0 & \forall\ t > t_1 \end{cases} ; \qquad \Phi(s) \overset{\text{def}}{=} X(t_1, s+h)B^*(s+h)$$

(iii) the function f satisfies the condition:

$$\sup\{f(t, x(\cdot), u(\cdot));\ t \in I\} \leq F(r) < \infty$$

for all $(x,u) \in B^n$ such that $||(x,u)||_2 \leq r(t_1 - t_o + \gamma)^{\frac{1}{2}}$ where $\lim_{r \to \infty} F(r)/r = 0$.

(iv) $\int_{t_o}^{t} \frac{ds}{\nu(s)} \geq \varepsilon > 0,\ C\mu/\lambda < 1,\ C > 0,\ \mu > 0,\ \lambda > 0$

Then the system (1) is Euclidean controllable.

REFERENCES

1. Manitius, A., "Necessary and Sufficient Conditions of Approximate Controllability for General Linear Retarded Systems", SIAM J. Control Optim. 19, 1981, pp. 516-532.

2. Chukwu, E.N., "On the Null-Controllability of Nonlinear Delay Systems with Restrained Controls", J. of Mathematical Analysis and Applications, 76, 1980, pp. 283-296.

3. Olbrot, A.W., "On Controllability of Linear Systems with Time Delays in Control", IEEE Trans. on Automatic Control, 1972, pp. 614-666.

4. Sinha, A.S.C., and Yokomoto, C.F., "Null-Controllability of a Nonlinear System with Variable Time Delay", IEEE Trans. on Automatic Control. Vol. AC-25, 1980, pp. 1234-1236.

5. Manitius, A. and Triggiani, R., "Function Space Controllability of Linear Retarded System, A derivation from Abstract Operator Conditions", SIAM J. Control Optim. 16, 1978, pp. 599-645.

NEAR-DISTURBANCE LOCALIZATION USING SECOND-ORDER MODES [*]

J.M. SANIUK and I.B. RHODES
Dept. of Electrical & Computer Engineering, University of California
Santa Barbara, CA 93106

ABSTRACT

Two scalar measures of input-output interaction are presented in the context of a generalized disturbance localization problem. In certain cases their gradients with respect to a feedback gain matrix have concise closed-form expressions and the F^* for which the gradient vanishes can be computed explicitly.

RESULTS

The product of the observability and reachability gramians of a linear system, denoted MW, is a matrix whose eigenvalues are non-negative and are invariant under changes of coordinates in the state space [4]. The *second-order modes* of the system are the square roots of these eigenvalues and can be interpreted as measures of mean-square energy throughput [3,6 and others]. The sum of the eigenvalues also arises in a natural way as a continuous-valued measure of interaction for purposes of nearly decoupling a system with several input and output channels: define measures of reachability and observability, respectively, on the state space by $R(x)=\frac{1}{2}x^T Wx$ and $O(x) = \frac{1}{2}x^T Mx$ [2]. Define a measure of interaction between the system inputs and outputs as

$$I = \underset{R^*(x)=1}{E} O(x) \tag{1}$$

where $R^*(x)$ denotes the conjugate functional $\frac{1}{2}x^T W^{-1}x$, and where the expectation is with respect to the uniform density on $\{R^*(x)=1\}$. It is readily seen (as in [7]) that $I = \frac{1}{n}$ tr MW. For convenience, we use the measure tr MW.

We present results for near-disturbance localization in discrete-time systems based on satisfaction of a first-order necessary condition. Consider a system

$$x_{k+1} = A\,x_k + B\,u_k + D\,v_k, \tag{2a}$$

$$y_k = C\,x_k \tag{2b}$$

and consider u_k as a local control input, v_k a disturbance. Define the problem of near-disturbance localization by state feedback as: find F = arg min tr $M_F W_{D,F}$, where $(\cdot)_F$ signifies that the system matrix is (A+BF), and where $W_{D,F}$ denotes the reachability matrix of ((A+BF),D). Consider the case of stable A_F. For the infinite-horizon problem in which M_F, $W_{D,F}$ are steady-state gramians, the gradient of tr $M_F W_{D,F}$ w. r. t. the feedback gain matrix F has a concise closed-form expression:

$$\frac{\partial}{\partial F} tr\ M_F W_{D,F} = 2B^T\left(M_F A_F \Gamma_{D,F} + \Lambda_F A_F W_{D,F}\right) \tag{3}$$

where Γ and Λ are *weighted gramians* defined by the steady-state discrete-time Lyapunov equations

$$\Gamma_{D,F} = A_F \Gamma_{D,F} A_F{}^T + W_{D,F} \tag{4}$$

$$\Lambda_F = A_F{}^T \Lambda_F A_F + M_F. \tag{5}$$

We wish to find F s. t. $\frac{\partial}{\partial F}$ tr $M_F W_{D,F}=0$.

Let p be the number of outputs. Consider first the case in which the first nonzero Markov parameter of the local system has full rank: $\rho(CB) = \cdots = \rho(CA^{\delta-2}B) = 0$ and $\rho(CA^{\delta-1}B) = p$. (This is always true of single-output systems.) If there are at least as many control inputs as outputs, with B having full (column) rank, the first-order

---

[*] Research supported in part by ONR Contract no. N00014-79-C-0549.

necessary condition has a solution that is independent of D; $F^* = -(CA^{\delta-1}B)^+ CA^{\delta}$, where $(\cdot)^+$ denotes any right inverse. This $F^* \in F(\mathbf{V}^*)$ (i.e., $(A + B F^*)\mathbf{V}^* \subseteq \mathbf{V}^*$), $\mathbf{V}^*$ denoting the supremal (A,B)-invariant subspace contained in the nullspace of C, and the map induced by $(A+BF^*)$ on $\mathbf{X}/\mathbf{V}^*$ has all its eigenvalues at the origin. This is a characterization of F^* for SISO systems. Furthermore, if the system is minimum-phase, F^* is the same as the optimal F obtained by the "cheap control" approach of finding

$$F^o = \lim_{\varepsilon \to 0} F_\varepsilon{}^o , \tag{6}$$

where

$$u_k = F_\varepsilon{}^o x_k = arg\ min. \left\{ \sum_{k=0}^{\infty} x_k{}^T C^T C x_k + \varepsilon u_k{}^T u_k \right\}. \tag{7}$$

This is equivalent to finding F^o = arg min. tr $C\, W_{D,F}\, C^T$ in the case of a minimum-phase system; the gradient of this latter measure is given by

$$\frac{\partial}{\partial F} tr\ C\, W_{D,F}\, C^T = 2B^T (M_F A_F W_{D,F}). \tag{8}$$

$F^* = -(CA^{\delta-1}B)^+ CA^{\delta}$ satisfies both first-order necessary conditions; in fact, it satisfies the stronger condition

$$(C(A+BF^*)^{k-1}B)^T (C(A+BF^*)^k) = 0,\ \ k=1,2,..., \tag{9}$$

which ensures that both gradients vanish term-by-term.

In the nonuniform rank case in which m $\geq$p>1 and CB =...=$CA^{\delta-2}$B =0, $CA^{\delta-1}$B $\neq$0 and $\rho(CA^{\delta-1}B)<$ p, one plausible extension of the above results is that F^* be characterized by (9), and further described by: 1) $F^* \in F(\mathbf{V}^*)$; 2) the map induced by $(A+BF^*)$ on $\mathbf{X}/\mathbf{V}^*$ has all eigenvalues at the origin; and 3) $F^* = -(B^T P B)^{-1} B^T P A$, where P is a (not necessarily stabilizing) solution to the Riccati equation $P = A^T P A + C^T C - A^T P B (B^T P B)^{-1} B^T P A$. The simulations performed so far support this conjecture.

REFERENCES

[1] W.M. Wonham, *Linear Multivariable Control: A Geometric Approach* (2nd ed.), New York: Springer-Verlag, 1979.

[2] I.B. Rhodes, "Some quantitaitve measures of controllability and observability and their implications," in *Proc. 8th IFAC Congress*, Kyoto, Japan, 1981, pp.30-35.

[3] B.C. Moore, "Principal component analysis in linear systems: controllability, observability, and model reduction," *IEEE Trans. Automat. Contr.* vol. AC-26, pp. 17-32, February 1981.

[4] C.T. Mullis and R.A. Roberts, "Synthesis of minimum roundoff noise fixed point digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-23, pp. 551-562, September 1976.

[5] H.Kwakernaak and R.Sivan, "The maximally achievable accuracy of linear optimal regulators and linear optimal filters," *IEEE Trans. Automat. Contr.*, vol. AC-17,no.1,Febr. 1972, pp.79-86.

[6] E. A. Jonckheere and L.M.Silverman, "A new set of invariants for linear systems -- application to reduced order compensator design," *IEEE Trans. Automat. Contr.*, vol. AC-28, no.10, Oct. 1983, pp.953-964.

[7] M. Athans and E.Tse, "A Direct Derivation of the Optimal Linear Filter Using the Maximum Principle", *IEEE Trans. Automat. Contr.*, vol. AC-12,no.6,Dec.1967, pp.690-698.

[8] H.Kwakernaak and R. Sivan, *Linear Optimal Control Systems*. New York: Wiley, 1972.

APPLICATION OF THE MULTIVARIABLE NYQUIST CRITERION TO DISTRIBUTED SYSTEMS

D. W. LUSE
Department of Electrical Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

ABSTRACT

This paper presents a Nyquist-type test which is useful for determining the closed-loop L^2 stability for a wide class of distributed multivariable systems. In particular, it can be applied to some systems whose open-loop transfer matrices are improper or have infinitely many right-half-plane poles.

A number of recent papers give generalizations of the multivariable Nyquist criterion of MacFarlane [1] to distributed systems [e.g. 2,3]. This author, however, found the hypotheses in these works to be too restrictive for certain applications to singularly perturbed systems, and to systems with an infinite number of pure imaginary poles. This paper describes a process by which the closed-loop stability of a wide class of systems can be tested through the use of elementary complex variable techniques.

Figure 1 shows the type of closed-loop system under consideration.

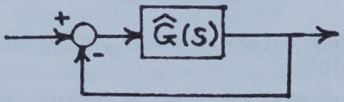


Figure 1

Here, $\hat{G}$ is the Laplace transform of the open-loop impulse response. Questions then arise, such as to what general class $\hat{G}$ may belong, and how the usual encirclement criterion must be modified. In [3], $\hat{G}$ is assumed at one point to be open-loop stable. The limit $\hat{G}(\infty)$ must be well defined to apply the results of [2]. The contention here is that with only slight weakening of results, $\hat{G}$ can belong to a very wide class of systems. The notation in the following is borrowed from [4].

We define the class of functions $\hat{g}(s)$ which are allowable as terms of $\hat{G}(s)$ in Figure 1.

Definition 1: Let T be the set of all $g \ \varepsilon \ \mathcal{D}'_+$ for which there exists σ_1 such that

$$ge^{-\sigma t} \ \varepsilon \ \mathcal{S}' \text{ for every } \sigma > \sigma_1$$

We also define T^{nxm} as the set of n x m matrices whose terms are members of T. The terms of $\hat{G}(s)$ are restricted to be in $\hat{T}$, the set of transforms of elements of T.

The following key result, [3, Thm. 2.1] gives a complete frequency domain characterization of L_2 stability. It is rephrased in the present context as the following theorem.

Theorem 1: An element H of T^{nxn} represents a finite gain L_2 stable convolution operator if and only if its Laplace transform $\hat{H}$ is analytic and uniformly bounded in the open right half plane.

The above theorem, together with the notion of the "local characteristic polynomial" of a transfer matrix yield the following Nyquist-type test. Let C^+ denote the open right half plane.

Theorem 2: Let G be a member of T^{nxn} and let $\hat{G}$ be its Laplace transform. Let Γ_i, $1 \le i < \infty$, be a sequence of simple, closed, piecewise smooth curves with interiors U_i. Suppose $C^+ \subseteq \bigcup_{i=1}^{\infty} U_i$. Then the following conditions are sufficient to guarantee closed loop stability.

1. $\hat{G}$ is analytic on all Γ_i
2. $\hat{G}$ is meromorphic in all U_i
3. $\sup_i \sup_{s \in \Gamma_i} ||\hat{G}(I+\hat{G})^{-1}|| < \infty$, $||\bullet||$ is any norm.
4. The paths of the eigenvalues $\lambda(\hat{G}(\Gamma_i))$ have a total winding number about -1 equal to the negative of the number of poles of $\hat{G}$ in U_i, for each i.

REFERENCES

[1] A. G. J. MacFarlane and I. Postlethwaite, "The generalized Nyquist stability criterion and multivariable root loci," Int. J. Contr., vol. 25, pp. 81-127, 1977.

[2] C. A. Desoer and Y. -T. Wang, "On the Generalized Nyquist Stability Criterion," IEEE Trans. Automat. Contr., vol. AC-25, no. 2, April 1980.

[3] J. M. E. Valenca and C. J. Harris, "Nyquist criterion for input/output stability of multivariable systems," Int. J. Control, 1980, vol. 31, no. 5, pp. 917-935.

[4] E. J. Beltrami and M. R. Wohlers, Distributions and the Boundary Values of Analytic Functions, (New York: Academic Press), 1966.

ON CAUSAL INPUT-OUTPUT SYSTEMS

WILLIAM L. ROOT and DEMETRIOS SERAKOS
Aerospace Engineering Department
The University of Michigan
Ann Arbor, MI 48109-2140

ABSTRACT

A dynamical system structure is assigned to causal, time-invariant, input-output systems characterized abstractly in terms of fitted families of normed linear spaces, with input-output maps belonging to one of a particular family of Banach spaces. Under a stability condition, the resulting state space is metrized as a subset of a Banach space of mappings. Continuity properties of state are given, together with facts about the connectivity and cardinality of the state space. A quantity called pseudo-state is introduced; it applies also to nonstable systems, and is sometimes a legitimate state. Some properties of pseudo-state and its relation to state are given. Tapered-memory systems are defined and a result about a feedback connection of such systems is presented.

INTRODUCTION

The problems considered in this paper have to do with assigning a dynamical system structure, in the sense of the definition in [1] (involving state and state transition), to an abstractly characterized causal, time-invariant and in general nonlinear input-output system, and then determining properties of this dynamical structure. The work reported here is a continuation of that in [2].

The format used for the input and output spaces is that of fitted families of normed linear spaces (as in [2] and [3]). A new norm for input-output maps is used that allows consideration of a wider class of systems than in [2]. Unfortunately there is not space here to give full technical details of all these matters, but [2] provides an adequate background. A rough summary of the necessary background material is presented in the next section.

In the succeeding sections we first recover the results on continuity of state of [2] for the state space with the new norm. These results apply to stable systems. There is then a very brief discussion of some connectivity and cardinality properties of the state space. A "pseudo-state" is introduced, which is well-defined for nonstable systems and sometimes is actually a state. A standard linear system example is sketched in context to illustrate these ideas. In the last section systems with "tapered memory" (a condition of decaying memory) are introduced and a result on feedback connections of such systems is presented. Most of the proofs are omitted.

PRELIMINARIES

A causal system (Y,F,U) is, in the context of this paper, a causal mapping F from an input space U to an output space Y ($y = F(u)$), where U is a translation invariant subset of a normed linear space A of time functions, and Y is either a Banach space B or an expanded space B^e of time functions. A proper definition of the spaces A and B depends on the introduction of fitted families of seminorms and of the corresponding normed linear spaces; this is done in [2] and [3], and B^e is defined in [2]. Suffice it to say here that an extended fitted family of seminorms $N = \{\|\cdot\|_{s,t};\ -\infty \leq s \leq t \leq \infty,\ -\infty < t,\ s < \infty\}$ on a linear space $\mathcal{L}$ of time functions is a translation-invariant family of seminorms, arbitrary as long as some mild "fitting" conditions are satisfied, with the property that $\|f\|_{s,t}$, $s \leq t$, is determined only by the behavior of the function on $[s,t]$ (or on a subinterval thereof). Associated with these seminorms are normed linear spaces $A_{s,t}$ whose elements are equivalence classes of functions in $\mathcal{L}$ (for outputs we use $|||\cdot|||_{s,t}$ and $B_{s,t}$). We write $\|\cdot\|_t \triangleq \|\cdot\|_{-\infty,t}$, $\|\cdot\| \triangleq \|\cdot\|_{-\infty,\infty}$, $A_t \triangleq A_{-\infty,t}$ and $A \triangleq A_{-\infty,\infty}$; A is the so-called bounding space. The output spaces $B_{s,t}$, B_t and B are assumed to be complete. The expanded space B^e is the nontopologized linear space of functions $f \in \mathcal{L}$ for which $|||f|||_s < \infty$ for all $s < \infty$; $B \subset B^e$. For the cases where s or t take infinite values, the following relations hold (actually by definition):

$$\|u\|_t = \lim_{s \to -\infty} \|u\|_{s,t}$$

$$\|u\|_{s,\infty} = \sup_{t \geq s} \|u\|_{s,t} \quad , \quad \|u\| = \sup_t \|u\|_t \quad .$$

Translation to the left by τ is denoted by L_τ, and $R_\tau = L_{-\tau}$; L_τ is a bijection on A (or B) and preserves norm.

An input space U is a time-invariant subset of A; in general U need not be linear (see [2],[3]), but here, for convenience, it is always assumed to be a linear subspace (more about this below). The spaces $U_{s,t}$ and U_t are then the subspaces of $A_{s,t}$ and A_t induced by U. It is also assumed here (as in [2]) that F is a time-invariant causal mapping. This means:

i) $F = L_\tau F R_\tau$

ii) For any u, v ε U $\| u-v \|_t = 0$, it follows that $||| F(u)-F(v) |||_t = 0$, for all t.

We let $\tilde{F}_t : U_t \to Y_t = B_t$ be the mapping defined by $||| \tilde{F}_t(u)-F(u) |||_t = 0$, and put $F_t = L_t \tilde{F}_t R_t$. Thus $F_t : U_o \to Y_o$ determines the behavior of the system at time t. The maps F_t are to belong to a certain Banach space now to be defined.

Definition Let $F_t : U_t \to Y_t = B_t$ be continuous and such that, for some fixed nonnegative integer N,

$$\| F_t \| \triangleq \sup_{u \,\varepsilon\, U_t} \frac{||| F_t(u) |||_t}{1 + ||| u |||_t^N} = \sup_{u \,\varepsilon\, U_t} \frac{||| F(u) |||_t}{1 + \| u \|_t^N} < \infty \quad . \qquad (1)$$

The set of all F_t satisfying (1) is a Banach space under the usual linear operations, with norm as shown. This space is denoted $\mathbb{C}_N^\infty(-\infty, t]$. (See [4] for details; these spaces are a special case of spaces $\mathbb{C}_\beta^\infty$ introduced there.) An analogue of the above definition holds for $F : U \to B$; one simply removes all subscripts t in (1). The resulting space is denoted $\mathbb{C}_N^\infty(-\infty, \infty)$. Similarly, there is an analogous definition for mappings from any $U_{p,q}$ into $B_{r,s}$.

When N = 0 $\mathbb{C}_N^\infty$ is trivially equivalent to the well-known Banach space $\mathbb{C}^\infty$ of continuous operators with bounded range (used in [2],[3]). An advantage of $\mathbb{C}_N^\infty$, N ≠ 0, is that the system maps need not have bounded range; so the chief reason for emphasizing bounded input spaces in [2] is removed. Furthermore, if N ≠ 0, the norm of $\mathbb{C}_N^\infty$ is equivalent for linear operators to the usual norm.

In order to define state in what follows (as in [2]), it is necessary that U have the property that it permits splicing of inputs. Roughly speaking, if u, v ε U, the splice of u and v at time s, denoted $u_s \leftarrow\!\!\!+\!\!\!\rightarrow v_{s,\infty}$, is the element of U that agrees with u for time t ≤ s and with v for t > s.

As mentioned, the results in [2] can be recovered using the norms of $\mathbf{C}_N^{\infty}$ spaces, and these are stated here. The proofs are omitted except for Theorem 1, which as it appears in [2] contains an error. (We take this opportunity to correct a mistake in the Appendix of [2]. The extended fitted family does not in general satisfy condition (4); i.e., $\| g \|_{s,\infty} \leq \| g \|_{s,t} + \| g \|_{t,\infty}$, $s \leq t < \infty$, is not true in general. This inequality was used in Theorem 1 under circumstances where it was not valid). For simplicity, we now require U to be a linear subspace of A. Since F is causal it induces a map $F_o : U_o \to Y_o$.

Definition. A system (Y,F,U) satisfies condition (S) if:

(1) (Y,F,U) is causal and time-invariant.

(2) The space A, the bounding space for inputs, permits splicing.

(3) The map $F_o : U_o \to Y_o$ is bounded in the norm of $\mathbf{C}_N^{\infty}(U_o, Y_o)$, for some fixed nonnegative integer N, and is uniformly continuous.

Condition (S) as just stated differs from condition (S) in [2] only in that the application caused by the use of bounded input spaces has been eliminated and the norm for F_o is changed.

Lemma 1. If (Y,F,U) satisfies condition (S), F is actually a mapping from U into Y = B (not just into B^e). It is bounded in the norm of $\mathbf{C}_N^{\infty}(U,B)$ and is uniformly continuous. ||

Lemma 2. Under the same conditions as for Lemma 1, for arbitrary fixed $t \in R$ and arbitrary $u_t \in U_t$, there exists a bounded uniformly continuous mapping $\tilde{\xi}_t^u : U_{t,\infty} \to B_{t,\infty}$ such that for all $v_{t,\infty} \in U_{t,\infty}$, $\tilde{\xi}_t^u(v_{t,\infty})$ is the same element of $B_{t,\infty}$ as that determined by $F(u_t \mapsto v_{t,\infty})$. ||

Definition. The natural state (or simply state) for a system (Y,F,U) satisfying condition (S), for any time t and any $u_t \in U_t$, is defined to be the mapping $\xi_t^u : U_{o,\infty} \to B_{o,\infty}$ given by $\xi_t^u = L_t \tilde{\xi}_t^u R_t$.||

Let Ξ be the set of all states, as just defined. Clearly $\Xi \subset \mathbf{C}_N^{\infty}(U_{o,\infty}, B_{o,\infty})$. Also if $\xi \in \Xi$, ξ can be achieved at any time t with some u_t (by time-invariance); and in particular $\xi = \xi_o^u$ for some u_o. The set of all u_o such that $\xi = \xi_o^u$ is an equivalence class. Of course if ϕ is a 1 to 1 map of Ξ onto a set X, X can also be regarded as a state space. However, we choose to regard it as such only if ϕ is a homeomorphism.

Remark. The state ξ_s^u at time s and the input $v_{s,t}$, $s < t$, determine the state ξ_t^w, $w = u_s \mapsto v_{s,t}$, at time t; they also determine the future output $y_{s,\infty}$. By the definition of Ξ, no set of states that have these properties can have cardinality less than the cardinality of Ξ. ||

Theorem 1. The state at time t for a system that satisfies condition (S) is a uniformly continuous function of the input prior to time t. That is, the mapping $u_t \to \xi_t^u$ is uniformly continuous with respect to the topologies of A_t and $\mathfrak{C}_N^{\infty}(U_{0,\infty}, B_{0,\infty})$.

Proof. From the definitions, since it is sufficient to consider states at time 0, and since $\xi_0^u = \tilde{\xi}_0^u$, one has

$$\begin{aligned}
\| \xi_0^u - \xi_0^{u'} \| &\triangleq \sup_{v_{0,\infty} \in U_{0,\infty}} \frac{|||\xi_0^u(v_{0,\infty}) - \xi_0^{u'}(v_{0,\infty})|||_{0,\infty}}{1 + \|v_{0,\infty}\|_{0,\infty}^N} \\
&\le \sup_{v_{0,\infty}} |||\xi_0^u(v_{0,\infty}) - \xi_0^{u'}(v_{0,\infty})|||_{0,\infty} \\
&= \sup_{v_{0,\infty}} |||F(u_0 \longmapsto v_{0,\infty}) - F(u_0' \longmapsto v_{0,\infty})|||_{0,\infty} \\
&= \sup_{v_{0,\infty}} \sup_{s>0} |||\cdots|||_{0,s} \\
&= \sup_{s>0} \sup_{v_{0,\infty}} |||R_s F L_s(u_0 \longmapsto v_{0,\infty}) \\
&\qquad\qquad - R_s F L_s(u_0' \longmapsto v_{0,\infty})|||_{0,s} \\
&\le \sup_{s>0} \sup_{v_{0,\infty}} |||F_0 L_s(u_0 \longmapsto v_{0,\infty}) - F_0 L_s(u_0' \longmapsto v_{0,\infty})|||_0
\end{aligned}$$

Now, for any $s \ge t$ and any $v_{0,\infty} \in U_{0,\infty}$,

$$\begin{aligned}
&\|L_s(u_0 \longmapsto v_{0,\infty}) - L_s(u_0' \longmapsto v_{0,\infty})\|_0 \\
&= \|(u_0 \longmapsto v_{0,\infty}) - (u_0' \longmapsto v_{0,\infty})\|_s \\
&\le \|u_0 - u_0'\|_0 + \|v_{0,\infty} - v_{0,\infty}\|_{0,s} = \|u_0 - u_0'\|_0 .
\end{aligned}$$

Therefore, by the uniform continuity of F_0, the map $u_t \to \xi_t^u$ is uniformly continuous. ||

Definition. The state trajectory given $u \in U$ of a system with state ξ_t^u is the mapping $t \to \xi_t^u$. ||

Theorem 2. Let (Y,F,U) satisfy condition (S) and suppose the left-extended fitted family $\{A_{s,t}\}$ is shift-continuous. Then the state trajectory given u, for any $u \in U$, is continuous. ||

SOME STATE SPACE PROPERTIES

It can be shown that under reasonable restrictions the state space is a connected subset of $\mathfrak{C}_N^{\infty}(U_{o,\infty},B_{o,\infty})$, and this results in an implication about its cardinality.

Consider a causal input-output system (Y,F,U). If there exists a state ξ^o such that for every input u, $\xi_t^u = \xi^o$ for all $t \leq t_o$, where $t_o = t_o(u)$ is finite, then the state ξ^o is called the relaxed state of (Y,F,U).

Lemma 3. Assume the state trajectories of a system (Y,F,U) are continuous and that there exists a relaxed state. Then Ξ is a connected set. ||

This lemma gives significance to the following result.

Lemma 4. If Ξ is connected and contains two states ξ and ξ' such that $\|\xi\| \neq \|\xi'\|$, then it contains an uncountable number of states. ||

Now if $C \subset U_o$ is a compact set, the set of states Ξ_c, consisting of ξ_o^u with $u \in C$, is compact, since the map from U_o into $\mathfrak{C}_N^{\infty}(U_{o,\infty},B_{o,\infty})$ is continuous, and hence complete. From this the following can be proved.

Lemma 5. Assume Ξ_c is connected and contains more than one state. Then it contains an uncountable number of states. ||

PSEUDO-STATE

The definition of state in natural form rests on the assumption of requirement (3) of condition (S). This is a basic stability requirement, and indeed we shall say a system is stable or not according as requirement (3) is or is not satisfied (this is not to rule out other concepts of stability for causal systems). Thus, we are left with the problem of specifying the state for a nonstable system. The difficulty with introducing an expanded space $B_{o,\infty}^e$ to repace $B_{o,\infty}$ in the definition of Ξ is that there is then no topology (at least no immediate topology) for Ξ ; we do not consider such an approach here. What is done is to introduce a "pseudo-state," which is sometimes actually a state, and which always has some of the properties of state. Even in the nonstable case it is possible to continue to consider causal systems defined for all $t \in R$ and for convenience we shall do so, although not all fitted families $\{A_{s,t}\}$ are then appropriate.

One can take for $\mathcal{C}$ (the underlying space of input functions) the closure under L_t, $-\infty < t < \infty$, of a linear space of functions $f(\cdot)$ such that $f(t) = 0$ for $t < 0$. Then for each $u \in U$, $\| u \|_{t_o} = 0$ for some $t_o = t_o(u)$, and the system will not have "blown up" by any finite time t, even though it is nonstable. In this situation there will be a relaxed state, as defined earlier.

So now, for F a causal map from U into B^e, we require only condition (S'), as follows, instead of (S).

Definition. A system (Y,F,U) satisfies condition (S') if it satisfies (1) and (2) of condition (S) and also:

(3') The map F_0 is uniformly continuous, and $F_T(u_0 \longmapsto v_{0,T})$: $U_{0,T} \to B_{0,T}$ with fixed u_0 is bounded in $\mathcal{C}_N^{\infty}$, $0 < T < \infty$. ||

It is not possible to recover Lemma 1. However, there is an analogue of Lemma 2, which permits a definition of pseudo-state analogous to that of state.

Lemma 2'. Under the condition (S'), for arbitrary fixed $t \in R$, fixed $0 < T < \infty$ and arbitrary $v_{o,t} \in U_{o,t}$ there exists a bounded continuous mapping $\tilde{\sigma}_t^u : U_{t,t+T} \to B_{t,t+T}$ such that for all $v_{t,t+T} \in U_{t,t+T}$, $\tilde{\sigma}_t^u(v_{t,t+T})$ is the same element of $B_{t,t+T}$ as that determined by $F_{t+T}(u_t \longmapsto v_{t,t+T})$. ||

Definition. For any $0 < T < \infty$, the pseudo-state (T) for a system (Y,F,U) satisfying condition (S') for any time t and any $u_{o\,T} \in U_{o,T}$ is defined to be the mapping $\sigma_t^u : U_{o,T} \to B_{o,T}$ given by $\sigma_T^u = L_t \tilde{\sigma}_t^u R_t$. ||

Remark. Obvious modifications of Theorems 1 and 2 hold for a pseudo-state (T). Since under condition (S'), pseudo-state (T) may be defined for any $0 < T < \infty$, it is worth noting the following. Let $\alpha > T_2 > T_1$, where α is defined in [2], and let $\Xi'(T_2)$, $\Xi'(T_1)$ be pseudo-state spaces with T_2, T_1. Then there is a continuous map from $\Xi'(T_2)$ into $\Xi'(T_1)$. In general this map is not invertible. Also, if the stronger condition (S) holds, there is a map from Ξ into $\Xi'(T)$, for any T.

We note that a pseudo-state transition function does not in general exist. However, for some systems the pseudo-state (T) is a state for any T. An example is provided in the next section. Following is an example of a causal input-output system for which the pseudo-state is not a state for any finite T. ||

Example. Let U be the space $\mathcal{C}$ of real-valued piecewise continuous time functions with the fitted family of seminorms given by

$\| u \|_{s,t} \triangleq \sup_{s \leq \tau \leq t} |u(\tau)|$. Let Y = U. Define the map F : U → Y by

$$y(t) = [F(u_t)]_t = \sum_{k=1}^{\infty} \frac{1}{2^k} \int_{t-2^k+1}^{t-2^k+2} u(s)\, ds.$$

For this F, for any fixed T > 0, there exist u', u'' ε U such that $\| u'-u'' \|_{0,T} = 0$ and $||| y'-y'' |||_{0,T} = 0$, but $||| y'-y'' |||_{0,M} \neq 0$ for some $M < \infty$.$\|$

EXAMPLE: LINEAR DIFFERENTIAL SYSTEM

A simple example to illustrate some of the foregoing is provided by nth order linear time-invariant systems described by

$$\begin{aligned} x &= A\,x + B\,u \\ y &= x \end{aligned} \tag{2}$$

where x(t) is an n-dimensional real vector, u(t) is an m-dimensional real vector and A and B are constant matrices. We let U be a linear space of piecewise continuous functions with

$$\| u \|_{s,t} \triangleq \sup_{s \leq \tau \leq t} \; \max_{i=1,\cdots,m} \; |u^i(\tau)|$$

where u^i is the i'th component of u, and let Y be the space of continuous functions with the corresponding norm. We compare the state x(t) at time t (the customary definition of state) with the natural state (our definition) and with pseudo-state.

For any fixed $0 < T < \infty$, x(t) determines a pseudo-state (T), σ, at time t. We may as well take t = 0, and we have by the variation-of-parameters formula

$$\| \sigma \| = \sup_{v \,\varepsilon\, U_{0,T}} \frac{||| \sigma(v) |||_{0,T}}{1 + \| v \|^N_{0,T}} \tag{3}$$

$$= \sup_{v \,\varepsilon\, U_{0,T}} \frac{||| \exp At \cdot x(0) + \int_0^t \exp A(t-\tau) \cdot B\, v(\tau)\, d\tau |||_{0,T}}{1 + \| v \|^N_{0,T}} .$$

It is easily seen that the mapping $\psi_T : R^n \to \mathfrak{C}^{\infty}_N(U_{0,T}, Y_{0,T})$ that carries x into σ is a homeomorphism. Thus all the pseudo-state spaces Ξ_T are homeomorphic, for all T.

If one writes the formulas analogous to (3) for the natural state ξ, it may be seen that ξ is defined if and only if the system (2) is b.i.b.s. stable (in the usual sense). If it is b.i.b.s. stable, then there is a homeomorphism ψ : $R^n \to \mathfrak{C}_N(U_{0,\infty}, Y_{0,\infty})$ that carries x into ξ. Thus, for a b.i.b.s. system (2), any of the quantities x, σ, ξ may equally be taken as state variables; for a non b.i.b.s. system (2), there is no natural state, but one may reasonably regard the σ's as state variables.

TAPERED MEMORY SYSTEMS

The concept of a tapered memory system is introduced as a way of characterizing abstractly a large class of systems for which the memory decays with time. Sufficient conditions are given for a feedback connection of two systems to have tapered memory.

Tapered Spaces and Tapered Memory

Definition [3]. Let $\{A_{s,t}\}$, $-\infty \leq s \leq t < \infty$, but $-\infty < t$, be a left-extended fitted family of normed linear spaces. For any c > 0 and any t, let $G(c,t) = \{f \in A_t : \|f\|_\tau \leq c,\ \tau \leq t\}$. Then $\{A_{s,t}\}$ is tapered if for every $\varepsilon > 0$, $c > 0$ and $t \in R$ there is a positive number $\delta = \delta(\varepsilon, c)$ such that $\|f\|_s \leq \|f\|_{s-\delta,s} + \varepsilon$ for all $f \in G(c,t)$ and any $s \leq t$. ||

Definition. A causal input-output system (Y,F,U) has tapered memory if for any $\varepsilon > 0$, $c > 0$ and $t \in R$ there is a positive number $\delta = \delta(\varepsilon,c)$ such that if $(u-u') \in G(c,t)$ and $\|u-u'\|_{s-\delta,s} = 0$, then $|||F(u)-F(u')|||_s \leq \varepsilon$ for any $s \leq t$. ||

The following lemma is directly implied by these definitions.

Lemma 6. Let (Y,F,U) be a causal system with F_t uniformly continuous. Then if the input space U is tapered the system has tapered memory. ||

Tapered Memory for a Feedback Connection

We consider two causal input-output systems (Y_1, F_1, U_1) and (Y_2, F_2, U_2) connected to form a simple feedback system according to the equations

$$\begin{aligned} e_1 &= u_1 - F_2 e_2 \\ e_2 &= F_1 e_1 = y_1 \end{aligned} \tag{4}$$

where $e_1 \in U_1$, $e_2 \in U_2$, and $U_2 = Y_1$, $U_1 = Y_2$. We want to require of Y_1 and Y_2 only that they be expanded spaces, B_1^e and B_2^e, respectively; so the input spaces must also be only expanded spaces. However, F_1 and F_2 must be such that they induce mappings $F_{1t} : U_{1t} \to Y_{1t}$ and $F_{2t} : U_{2t} \to Y_{2t}$ for any finite t. A standard small gain theorem of nonlinear feedback systems, which we quote below from [5] with some minor changes, now applies.

Theorem [5]. For the feedback system of (4) if there are constants $\tilde{\gamma}_1, \tilde{\gamma}_2$

with $\tilde{\gamma}_1\tilde{\gamma}_2 < 1$ such that for all t and all $\eta_1, \eta_1' \varepsilon U_1$ and $\eta_2, \eta_2' \varepsilon U_2$

$$|||F_1\eta_1 - F_1\eta_1'|||_t \leq \tilde{\gamma}_1 \, ||\eta_1 - \eta_1'||_t$$

$$|||F_2\eta_2 - F_2\eta_2'|||_t \leq \tilde{\gamma}_2 \, ||\eta_2 - \eta_2'||_t$$

then: (i) For all $u_1 \varepsilon U_1$ there exists a unique solution for $e_1 \varepsilon U_1 = B_2^e$ and $e_2 \varepsilon U_2 = B_1^e$. (ii) The map $u_1 \to (e_1, e_2)$ is uniformly continuous with respect to the topologies of U_{1t} and $U_{1t} \times U_{2t}$ for all $t \varepsilon R$. (iii) If in addition the e_1 and e_2 corresponding to $u_1 = 0$ are in B_2 and B_1 respectively (the bounding spaces), then $u_1 \varepsilon B_2$ implies $e_1 \varepsilon B_2$, $e_2 \varepsilon B_1$. ||

We can now state a theorem that gives conditions for a feedback system to have tapered memory.

Theorem 3. Assume the conditions of the theorem quoted above hold, including (iii). Assume further that if given $G(c,t)$ and $\varepsilon > 0$ there exist $\delta = \delta(\varepsilon, c)$, $k_1 = k_1(\varepsilon, c)$, $k_2 = k_2(c)$ with $k_1 k_2 \leq K < 1$ (K does not depend on ε) such that

$$|||F_1\eta_1 - F_1\eta_1'|||_s \leq k_1 \, ||\eta_1 - \eta_1'||_{s-\delta,s} + \varepsilon \tag{5}$$

$$|||F_2\eta_2 - F_2\eta_2'|||_s \leq k_2 \, ||\eta_2 - \eta_2'||_s$$

for all $(\eta_1 - \eta_1') \varepsilon G(c,t)$, $(\eta_2 - \eta_2') \varepsilon G(\tilde{\gamma}_2 c, t)$.

Then,

$$|||y_1 - y_1'|||_s \leq \frac{k_1}{1-K} \, ||u_1 - u_1'||_{s-\delta,s} + \frac{\varepsilon}{K} . || \tag{6}$$

We note that condition (5) of the hypothesis is somewhat stronger than the tapered memory condition for the forward system, but also the conclusion (6) is stronger than tapered memory for the feedback system.

REFERENCES

[1] R.E. Kalman, P.L. Falb, M.A. Arbib, "Topics in Mathematical System Theory," McGraw-Hill (1969).

[2] W.L. Root, "A Note on State Trajectories of Causal Input-Output Systems," Proc. 21st Allerton Conf. on Communication, Control and Computing, (1983).

[3] W.L. Root, "Considerations Regarding Input and Output Spaces for Time-Varying Dynamical Systems," Appl. Math. Optim. 4: 365-384 (1978).

[4] W.L. Root, "The Identifiability of Abstract Systems," Tech. Report, in preparation, The University of Michigan.

[5] C.A. Desoer and M. Vidyasagar, Feedback Systems: Input-Output Properties, Academic Press (1975).

MINIMAL REALIZATIONS OF MATRIX FRACTION DESCRIPTIONS

SHOU-YUAN ZHANG
AGS Department
Brookhaven National Laboratory
Upton, New York 11973

ABSTRACT

In this paper, we present the systematic realizations for general matrix fraction description $G = PQ^{-1}R$, and for the matrix fraction PQ^{-1} and $Q^{-1}R$. The realizations are based on the Kalman-Fuhrmann realization theory and the dual theory. By choosing the proper basis for the state space of the system, the realizations appear to be very simple. The realizations of PQ^{-1} and $Q^{-1}R$ are just in a similar way to the realizations of single-variable systems.

I. INTRODUCTION

Many minimal realizations for matrix fraction description of linear dynamic systems has been proposed in last two decades. For single-variable systems, two well known realization forms are the controller canonical form and observer canonical form [1]. Usually, they are called controllable and observable canonical forms, see, e.g., [2.3]. In the forms, all nontrivial elements in the state space expression A, b, c, are all directly from the coefficients of the denominator and numerator of the transfer function of the system, as long as the denominator is monic. The forms are so nice that researchers have been persistently looking for the similar forms in the minimal realization for multivariable systems. This kind realizations are presented, for example, in [1]. We note that there are another two realization forms for single-variable systems, called the controllability and observability forms in [1], where the nontrivial elements in the matrix A are from the denominator of the transfer function, however, the nontrivial elements in b and c are from the Markov parameters. These forms are also developed to fit in the multivariable systems [1]. As expected, all forms in multivariable case are no longer as neat as the ones for single-variable case. They also lost the canonicallility. One may ask: Is there any realization for multivariable systems being more similar to the ones for single-variable systems? The answer is positive.

In [4], Kalman gave the linear dynamic systems with the module structure interpretation. Later, Fuhrmann [5,6] developed the realization theory given there. Three different fractions were considered, i.e., the systems are fractioned by $Q^{-1}R$, PQ^{-1}, and the general matrix fraction description $PQ^{-1}R$. In [7,8], the realization was further discussed.

The core of the Kalman-Fuhrmann realization is a Q-module over the s-polynomials. This module in practice is represented by a linear vector space -- the state space. The matrix A of the realization function as the shifting operator in the module, the matrix B maps the input space into the module and the matrix C maps the vectors in the module to the output space. We note that this realization theory provides an opportunity to build up a set of simple realization for multivariable systems. In addition, the realization are with obvious physical meanings.

In multivariable system, due to the different fraction, the numerical matrix can be sitting on one side, e.g., in $Q^{-1}R$ or PQ^{-1}, or, on both sides, in $PQ^{-1}R$. We call $PQ^{-1}R$ as the general matrix fraction description, and take $Q^{-1}R$ and PQ^{-1} as the special cases of $PQ^{-1}R$.

In this paper, we start from the general matrix function description $G(s) = PQ^{-1}R$. The Kalman-Fuhrmann realization regarding to this fraction will be discussed. Without loss of generality, we assume that Q is row reduced with the row degree coefficient matrix I. Since the Q-module itself is a vector space over a field, we build up a standard basis for the vector space according to the row degrees of Q. Thus, the shifting matrix A can be established in a similar way to the one in the observer form for single-variable systems. Accordingly, the matrix B can be set up with the coefficients in R, the matrix C can then be found from the Markov parameters of the subsystem PQ^{-1}.

If we let the denominator polynomial matrix Q be column reduced, a realization in the dual forms is easily deduced.

In both forms, we can let P or R be identity matrix. Thus, four realization forms based on $Q^{-1}R$ and PQ^{-1} can be named as controller, observer, controllability and observability forms. These forms are all very similar to the corresponding forms of single-variable systems.

In the following, we first discuss the minimal realization for the general matrix fraction description $PQ^{-1}R$, where Q is row reduced, by applying the Kalman-Fuhrmann realization theory. The dual case, where Q is column reduced will be briefly discussed. Then, the realizations for four special cases by PQ^{-1} and $Q^{-1}R$ are presented, which are probably more useful in practice. A conclusion follows.

II. THE MINIMAL REALIZATIONS FOR $PQ^{-1}R$

In this section, we present the Kalman-Fuhrmann realization theory for the general matrix fraction $PQ^{-1}R$. A proper basis for the state space of the realization will be chosen. A realization scheme will then be proposed and verified. The dual case will also be considered.

In the following text, F denotes a field, F^n denotes the n-triple vector space in F. Let $F[s]$ be the polynomial ring over the field F, and $F^n[s]$ be the set of polynomials with coefficients in F^n. Let $F^n(s)$ denote the rational functions with coefficients in F^n. A map π is defined such that $\pi F^n(s)$ is the strictly proper part of $F^n(s)$. Thus, an arbitrary n-tuple polynomial vector q, $q \in F^n(s)$, can be written as

$$g = \pi g + \tilde{g}$$

where $\tilde{g} \in F^n[s]$.

Consider a given q x q nonsingular polynomial matrix $Q(s)$. The polynomial vectors f in $F^q[s]$, such that $Q^{-1}f$ is strictly proper, construct a vector space K_Q. In particular, if Q is row reduced, K_Q is spanned by all q-tuple polynomial vectors with the row degrees less than the corresponding row degrees of Q. As shown by Kalman [4], K_Q is a module over $F^q[s]$. Now, the map π_Q is defined for the q-tuple polynomials by

$$\pi_Q f = \pi Q(Q^{-1}f), \quad f \in F^q[s]$$

i.e., for any q-tuple polynomial vector f, $\pi_Q f$ is the part of f, which belongs to K_Q.

Let the general matrix fraction of the p x r strictly proper system G be

$$G = PQ^{-1}R \tag{1}$$

where Q is a q x q nonsingular polynomial matrix. The Kalman-Fuhrmann realization $\{A, B, C\}$ can be shown as follows:

$$\begin{aligned} B:&\quad F^r \to K_Q \;;\quad u \mapsto \pi_Q(Ru) \\ A:&\quad K_Q \to K_Q \;;\quad f \mapsto \pi_Q(sf) \\ C:&\quad K_Q \to F^p \;;\quad f \mapsto (PQ^{-1}f)_1 \end{aligned} \tag{2}$$

where u is the input, and $(PQ^{-1}f)_1$, is from the Markov parameters of $PQ^{-1}f$, i.e., from

$$PQ^{-1}f = \Sigma_1^\infty (PQ^{-1}f)_i s^{-i}$$

To accomplish the realization, we need first to choose a basis for the vector space of K_Q, and furthermore, to express the basis in a constant vector form. Consider the general matrix fraction of (1), without loss of generality, we assume that Q is row reduced with row degrees m_i, i = 1, ..., q, and the row degree coefficient matrix of Q is an identity matrix. Thus, all vectors in K_Q must meet the requirement: their ith row degree should be less than m_i, i = 1, ..., q. Thus, a natural basis for K_Q can be chosen as follows

$$\begin{bmatrix} s^{k_1} \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} ; \begin{bmatrix} 0 \\ s^{k_2} \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} ; \ldots ; \begin{bmatrix} 0 \\ 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ s^{k_q} \end{bmatrix} \tag{3}$$

where

$$0 \leq k_i \leq m_i - 1 \;, \quad i = 1, \ldots, q$$

In a strictly proper system $G = PQ^{-1}R$, the row degrees of R are not necessarily less than the corresponding row degrees of Q, hence, we perform the operation π_Q on R, and have

$$\hat{R} = \pi_Q R$$

Then, we write

$$Q = \mathrm{diag}\{s^{m_i}\} + \mathrm{diag}\{s^{m_i - 1}\}Q_1 + \ldots + \mathrm{diag}\{s^{m_i - m}\}Q_m \tag{4}$$

$$\hat{R} = \mathrm{diag}\{s^{m_i-1}\}\hat{R}_1 + \mathrm{diag}\{s^{m_i-2}\}\hat{R}_1 + \dots + \mathrm{diag}\{s^{m_i-m}\}\hat{R}_m \qquad (5)$$

where

$$m = \max\{m_i\}$$

is the observability index of G, and

$$\mathrm{diag}\{s^{m_i-j}\} = \begin{bmatrix} s^{m_1-j} & & \\ & s^{m_2-j} & \\ & & \ddots \; s^{m_q-j} \end{bmatrix} , \; j = 0, 1, \dots, m$$

Since $m \geq m_i$, for all i, some elements in the diagonal matrices

$$\mathrm{diag}\{s^{m_i-j}\} \; , \; 2 \leq j \leq m$$

may have the negative s-powers. These elements are used to keep a neat realization form, and the corresponding state space, called the redundant state space, will be deleted in the final step of the realization. Now, the realization matrices A and B can be shown as follows:

$$A = \begin{bmatrix} 0 & & -Q_m \\ I & 0 & -Q_{m-1} \\ & \ddots & \vdots \\ & I & -Q_1 \end{bmatrix} \; ; \; B = \begin{bmatrix} \hat{R}_m \\ R_{m-1} \\ \vdots \\ \hat{R}_1 \end{bmatrix} \qquad (6)$$

Since all row degrees of $\hat{R}$ are less than the corresponding row degrees of Q, for every u in F^r, Ru belongs to K_Q. Thus, we choose B just according to the basis shown in (2), in the sequence of (6). In the following, we show the shifting function of A. Let f belong to K_Q. Let $f' = sf$. The function of A operated on f should yield $\pi_Q f'$. We denote the column expression of the polynomial vector, for example, f, according to the basis shown in (6), by $\bar{f}$. The column expression of $\pi_Q f'$ can be found by

$$(\bar{Q} \; \bar{f}')U = \begin{bmatrix} Q_m & \\ Q_{m-1} & f_m \\ \vdots & \vdots \\ Q_1 & f_2 \\ I & f_1 \end{bmatrix} \begin{bmatrix} I & -f_1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} Q_m & -Q_m f_1 \;, \; m \\ Q_{m-1} & -Q_{m-1} f_1 + f_m \\ \vdots & \vdots \\ Q_1 & -Q_1 f_1' + f_2' \\ I & 0 \end{bmatrix} = \left[\bar{Q} \; \overline{\pi_Q f'} \right]$$

But we have

$$A\bar{f} = \begin{bmatrix} 0 & & -Q_m \\ I & & -Q_{m-1} \\ & \ddots & \vdots \\ & I & -Q_1 \end{bmatrix} \begin{bmatrix} f_m \\ f_{m-1} \\ \vdots \\ f_1 \end{bmatrix} = \begin{bmatrix} -Q_m f_1 \\ -Q_{m-1} f_1 + f_m \\ \vdots \\ -Q_1 f_1 + f_2 \end{bmatrix} = \overline{\pi_Q f'}$$

Thus, the shifting function of A is proved.

The matrix C of the realization should be determined by the Markov parameters of the subsystem PQ^{-1}. We have

$$PQ^{-1} = W = \left(W_{11} \ldots W_{1q}\right)s^{-1} + \left(W_{21} \ldots W_{2q}\right)s^{-2} + \ldots$$
$$+ \left(W_{m_q 1}, \ldots W_{m_q q}\right)s^{-m_q} + \ldots$$

where w_{ij}'s are the p x 1 vectors, i = 1, 2, ... , j = 1, 2, ... q.

By using the same basis as that of (6), we write the q-tuple polynomial vector, $f \in K_Q$, by

$$f = \left[\begin{array}{c|cccc|cccc} & s^{m_1-2} & & & & s^{m_1-1} & & & \\ \cdot\ \cdot\ \cdot & & s^{m_2-2} & & & & s^{m_2-1} & & \\ & & & \ddots & & & & \ddots & \\ & & & & s^{m_1-2} & & & & s^{m_q-1} \end{array}\right] \quad \bar{f}$$

Thus,

$$(PQ^{-1}f)_1 = \lim_{s\to\infty} \pi s PQ^{-1} f = \lim_{s\to\infty} \pi s W f$$

$$= \lim_{s\to\infty} \pi s\left\{\left(W_{11} \ldots W_{1q}\right)s^{-1} + \ldots + \left(W_{m_q 1}, \ldots W_{m_q q}\right)s^{-m_q} + \ldots\right\}$$

$$\left[\begin{array}{c|cccc|cccc} & s^{m_1-2} & & & & s^{m_1-1} & & & \\ \cdot\ \cdot\ \cdot & & s^{m_2-2} & & & & s^{m^{2-1}} & & \\ & & & \ddots & & & & \ddots & \\ & & & & s^{m_q-2} & & & & s^{m_q-1} \end{array}\right] \quad \bar{f}$$

$$= \lim_{s\to\infty} \pi\left\{\left(W_{11} \ldots W_{1q}\right)s^{-1} + \ldots + \left(W_{m_q 1} \ldots W_{m_q q}\right)s^{-m_q} + \ldots\right\}$$

$$\left[\begin{array}{c|cccc|cccc} & s^{m_1-1} & & & & s^{m_1} & & & \\ \cdot\ \cdot\ \cdot & & s^{m_2-1} & & & & s^{m^2} & & \\ & & & \ddots & & & & \ddots & \\ & & & & s^{m_q-1} & & & & s^{m_q} \end{array}\right] \quad \bar{f}$$

$$= \left(\ldots \ \ldots \ W_{(m_1-1)1} \ \ldots \ W_{(m_q-1)q} \ W_{m_1 1}, \ \ldots \ W_{m_q q}\right)\bar{f}$$

$$= \bar{W}\ \bar{f}$$

Hence, we have

$$C = \bar{W} \qquad (6')$$

An interpretation is due at this point for the realization (6,6'). The matrix A plays an important role in the realization. First of all, the matrix A defines the state space of the realization, according to the basis shown in (6), which is in fact from the row degrees of Q. Secondly, for any f belonging to K_Q, the left multiplication of A is a one step shift as can be seen from (7). The matrix C picks up the one step delayed output from the vector in K_Q. An arbitrary input from the input space is mapped by B into the state space of the realization. Thus, CBu is the one step delayed putput due to the input u. Since ABu is the one step shifted input, then, CABu is the two step shifted output, and so on for all CA^iBu, i = 1, ... Hence, the dynamic system can be expressed by the Markov parameters consisted of the matrices of the realization, i.e.,

$$G = PQ^{-1}R = CBs^{-1} + CABs^{-1} + \ldots + CA^{i}Bs^{-i-1} + \ldots$$

To achieve the minimal realization, the redundant state space in A, i.e., those corresponding to the basis with negative s-powers, must be deleted. At the same time, the same rows and columns in B and C, respectively, have also to be deleted.

We show an example to illustrate the realization procedure.

Example 1:

Consider the strictly proper system

$$G = PQ^{-1}R = \begin{bmatrix} 0 & s-1 \\ 1 & s+1 \end{bmatrix} \begin{bmatrix} s^2+1 & s \\ s^2 & s^3+1 \end{bmatrix}^{-1} \begin{bmatrix} s & 0 \\ 1 & s-1 \end{bmatrix}$$

We note that the denominator matrix Q is row reduced with row degree coefficient matrix I. Two row degrees of Q are 2 and 3, respectively. Thus, we write

$$Q = \operatorname{diag}\{s^2, s^3\} + \operatorname{diag}\{s, s^2\}\, Q_1 + \operatorname{diag}\{1, s\}\, Q_2 + \operatorname{diag}\{s^{-1}, 1\}\, Q_3$$

$$= \begin{bmatrix} s^2 & \\ & s^3 \end{bmatrix} + \begin{bmatrix} s & \\ & s^2 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & \\ & s \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} s^{-1} & \\ & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Since the row degrees of R are less than the corresponding row degrees of Q, we can write

$$R = \operatorname{diag}\{s, s^2\}\, R_1 + \operatorname{diag}\{1, s\}\, R_2 + \operatorname{diag}\{s^{-1}, 1\}\, R_3$$

$$= \begin{bmatrix} s & \\ & s^2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 1 & \\ & s \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} s^{-1} & \\ & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix}$$

We write the realization matrices A and B, which are not necessarily minimal

$$A = \begin{array}{l} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{bmatrix} \\ \;\;\uparrow \end{array} \begin{array}{l} \leftarrow \\ \\ \\ \\ \\ \\ \\ \end{array} \quad \begin{array}{c} \leftarrow s^{-1} \rightarrow \\ \leftarrow 1 \rightarrow \\ \leftarrow 1 \rightarrow \\ \leftarrow s \rightarrow \\ \leftarrow s^{2} \rightarrow \\ \leftarrow s \rightarrow \\ \\ \end{array} \qquad B = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{array}{l} \leftarrow \\ \\ \\ \\ \\ \\ \end{array}$$

To find C, we have

$$PQ^{-1} = W = (W_{11}\ W_{12})s^{-1} + (W_{21}\ W_{22})s^{-2} + (W_{31}\ W_{32})s^{-3} + \ldots$$

$$= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} s^{-1} + \begin{bmatrix} -1 & 1 \\ 0 & 1 \end{bmatrix} s^{-2} + \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} s^{-3} + \ldots$$

Thus,

$$C = W = \left(W_{01}\ W_{12} W_{11}\ W_{22}\ W_{21}\ W_{32} \right) = \begin{bmatrix} \overset{\downarrow}{0} & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

After deleting the redundant state space, which is indicated in A, B, C, we have the minimal realization $\{\hat{A},\hat{B},\hat{C}\}$:

$$\hat{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \end{bmatrix} ; \hat{B} = \begin{bmatrix} 1 & -1 \\ 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{bmatrix} ; \hat{C} = \begin{bmatrix} 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

In the next, we discuss the dual realization of the one of (6,6'). The system is also fractioned as that in (1). The only difference is that the denominator matrix Q is column reduced, not necessarily row reduced.

Let the column degree coefficient matrix of Q be I, and the ith column degree be m_i. $i = 1, 2, \ldots, q$.

$$Q = \text{diag}\{s^{m_i}\} + Q_1 \text{diag}\{s^{m_i-1}\} + \ldots + Q_m \text{diag}\{s^{m_i-m}\} \qquad (8)$$

In a similar way to the realization presented above, we find $\hat{P}$ from P, such that the column of $\hat{P}$ are less than the corresponding column degrees of Q, and write $\hat{P}$ in the same way to (8). Thus, the realization matrices A and C are

$$A = \begin{bmatrix} 0 & I & & \\ & & \ddots & \\ & & & I \\ -Q_m & -Q_{m-1} & \cdots & -Q_1 \end{bmatrix} ; \quad C = \begin{bmatrix} \hat{P}_m & \hat{P}_{m-1} & \cdots & \hat{P}_1 \end{bmatrix}$$

We also can find the matrix B from the Markov parameters of the subsystem $Q^{-1}R$ in a similar way to the one in (6').

It is interesting to note that the Kalman-Fuhrmann theory cannot be applied to the dual realization, and a further study is believed to be necessary. However, the realization may be interpreted in another point of view. It is noticed that the Kalman theory for the dynamic systems is in essential as same as that called Nerode equivalence. See, for example, [1], where the state space of the dynamic system is identified from the input space. From this point of view, the dual realization can be interpreted in this way : the state space of the system is identified from the output space.

III. FOUR REALIZATIONS FOR PQ^{-1} and $Q^{-1}R$

In Section II, we presented two realizations for $PQ^{-1}R$. Although the fraction $PQ^{-1}R$ is a general one, in practice, the matrix fraction description PQ^{-1} and $Q^{-1}R$ may be more often used. In this section, we present the realization for PQ^{-1} and $Q^{-1}R$. The realization for the left matrix fraction $Q^{-1}R$, where Q is row reduced, was first proposed in [9]. The realization is in a very similar form to the canonical observable form of single-variable systems, and all nontrivial elements in the realization are directly from the coefficients of Q and R.

We assume that Q is row reduced with row degrees m_i, $i = 1, 2, \ldots q$. The row degree coefficient matrix is I. Thus, Q can be written as in (4). Since $Q^{-1}R$ is strictly proper, the row degrees of R must be less than the corresponding row degrees of Q. Hence, we write R by (5). The realization is simply as

$$A = \begin{bmatrix} 0 & & & -Q_m \\ I & & & -Q_{m-1} \\ & \ddots & & \vdots \\ & & I & -Q_1 \end{bmatrix} ; \; B = \begin{bmatrix} R_m \\ R_{m-1} \\ \vdots \\ R_1 \end{bmatrix} ; \; C = \begin{bmatrix} 0 & \ldots & 0 & I \end{bmatrix} \tag{9}$$

where C is from

$$C\bar{f} = (Q^{-1}f)_1 = \lim_{s\to\infty} sQ^{-1}f$$

$$= \lim_{s\to\infty} Q^{-1} \left[\begin{array}{ccc|ccc} \ldots & s^{m_1-1} & & s^{m_1} & & \\ & & \ddots & & \ddots & \\ & & s^{m_q-1} & & & s^{m_q} \end{array} \right] \bar{f}$$

$$= \lim Q^{-1} \operatorname{diag}\{s^{m_i}\} \left(\ldots \mid s^{-1} I_q \mid I_q \right) \bar{f}$$

$$= \left(0 \ \ldots \ 0 \ I \right) \bar{f}$$

The observability of the realization can be easily verified by using the form (9). If Q and R are left coprime, the controllability of the realization can also be easily verified by using the existing result of the resultant matrix, see [9].

The redundant state space can always be deleted at the last step, even some operations of the matrices A, B, C are involved. We note that the operation are easier if we use the neat form in (9). All rows in O_i, R_i, $i = 1, 2, \ldots q$, and columns in C, corresponding to the redundant state are identically zero, Thus, the operations by using $\{A, B, C\}$ and the minimal set $\{\hat{A}, \hat{B}, \hat{C}\}$ are with no substantial difference.

Following [1], we call the realization to be in observer form. The controller form is ready for the fraction PQ^{-1} with Q column reduced, which can be taken as the dual case of the controller form realization. In particular, it can be written as

$$A = \begin{bmatrix} 0 & I & & \\ & & \ddots & \\ & & & I \\ -Q_m & -Q_{m-1} & \ldots & -Q_1 \end{bmatrix} ; \; B = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ I \end{bmatrix} ; \; C = \left(P_m \ P_{m-1} \ \ldots \ P_1 \right)$$

Consider the fraction PQ^{-1}, where Q is row reduced with row degrees m_i, $i = 1, 2, \ldots q$, the row degree coefficient matrix of Q is I. Write Q in the form of (4). Note that in this case, PQ^{-1} can be seen as the special case of $PQ^{-1}R$, Q is row reduced, and $R = I$. The matrices A and

B are written as that of (6). The matrix B consists of 0's and an I, with some row permutations. In particular, the 1 in the ith column of B should be in the row corresponding to the zero s-power basis of ith row of Q. The matrix C is written out by the Markov parameters of PQ^{-1}. The following example illustrates the realization.

Example 2:

We take the left part of the system used in Example 1, i.e.,

$$PQ^{-1} = \begin{bmatrix} 0 & s-1 \\ 1 & s+1 \end{bmatrix} \begin{bmatrix} s^2+1 & s \\ s^2 & s^3+1 \end{bmatrix}^{-1}$$

The realization then is

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{bmatrix} ; \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} ;$$

$$C = \begin{bmatrix} 0 & 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

After deleting the redundant state, we have a minimal realization.

Since the Markov parameters used in the realization can be obtained from the impulse response of the system, and are always recorded and used in the identification of transfer function, this realization is easy to use in practice.

Also following [1], we call the realization in the controllability form.

It is of interest to see that in PQ^{-1}, when Q is column reduced, the strict properness can be insured by the less column degrees of P, hence, the controller realization simply uses the coefficients in Q and P; when Q is row reduced, no rule can be found to insure the strict properness from the row or column degrees of P, thus, the matrix C can no longer be found from the coefficients of P, instead, it is found from the Markov parameters.

This realization is also, in fact, identical with the Nerode realization [1], if we disregard some row and column permutations. However, it is in a simpler and more systematic way.

The observability realization form for $Q^{-1}R$, where Q is column reduced, is easy to find in a similar way.

IV. CONCLUSION

In this paper, we presented the systematic realizations for $PQ^{-1}R$, PQ^{-1} amd $Q^{-1}R$, based on the Kalman-Fuhrmann realization theory and the dual theory. By choosing the proper basis, the realizations appear to be very simple. The realization of PQ^{-1} and $Q^{-1}R$ can be taken as the extensions of the realization for single-variable system.

It is believed that the realizations presented in this paper are beneficial to the better understanding of the dynamical systems, and the relations of state space description and matrix fraction description of the systems.

REFERENCES

(1) W.A. Wolovich, Linear Multivariable Systems, New York: Springer-Verlag, 1974.

(2) T. Kailath, Linear Systems, Englewood-Cliffs, New Jersey: Prentice Hall, 1980.

(3) C.T. Chen, Linear System Theory and Design, New York: Holt, Rainhard and Winston, 1984.

(4) R.E. Kalman, P.L. Falb, and M.A. Arbib, Topics in Mathematical System Theory, New York: McGraw-Hill, 1969.

(5) P.A. Fuhrmann, "Algebraic System Theory: An Analyst's Point of View," J. Franklin Inst., vol. 301, pp. 521-540, 1976.

(6) P.A. Fuhrmann, "On Strict System Equivalence and Similarity," Int. J. Contr., vol. 25, pp. 5-12, 1977.

(7) M.L.J. Hautas and M. Heymenn, "Linear Feedback -- An Algebraic Approach," SIAM J. Contr. Optimiz., vol. 10, pp. 83-105, 1978.

(8) E. Emre, "On a Natural Realization of Matrix Fraction Descriptions," IEEE Trans. Automat. Contr., vol. AC-25, pp. 288-289, 1980.

(9) S.Y. Zhang, "An Irreducible Realization Scheme for Matrix Fractions of Multivariable systems," Systems & Control Letters, vol. 5, No. 6, 1985.

ALGORITHMS FOR THE MINIMUM K-PARTITIONING OF RECTILINEAR POLYGONS

OLIVER GUNTHER
Computer Science Division
570 Evans Hall
University of California at Berkeley
Berkeley, CA 94720
E-Mail: oliver@ucbingres.arpa

ABSTRACT

This paper is about rectilinear partitions that decompose a rectilinear polygon with n vertices into components that have no more than k vertices each, where k is given and $k < n$. For two classes of rectilinear polygons we present partitioning algorithms that yield a minimum number of components. The first algorithm is for non-degenerate rectilinear polygons that are x-convex or y-convex. It has time complexity $O(n)$. For degenerate polygons that are x-convex *and* y-convex an $O(n^4)$ algorithm is presented.

1. Introduction

A *rectilinear polygon* is a simple polygon without holes with the additional constraint that all of its sides are either parallel or perpendicular to each other. An *n-gon* is a rectilinear polygon with n vertices. A rectilinear polygon is called *x-convex* if the intersection of each horizontal line with the polygon is either one line segment or empty. Analogously, *y-convexity* is defined. A vertex of a rectilinear polygon is called *convex* if its interior angle is 90°, and *concave* if its interior angle is 270°. A *cut* of a rectilinear polygon is a horizontal or vertical line segment inside the polygon that joins two points of the polygon boundary. A cut is called *non-degenerate* if exactly one of its endpoints is a concave vertex, and *degenerate* if both are. A rectilinear polygon P is called *non-degenerate* if it does not contain any degenerate cuts, otherwise it is called *degenerate*. A *rectilinear partition* of a rectilinear polygon P is a set of disjoint rectilinear polygons whose edges are parallel to the edges of P and whose union is P.

This paper is about rectilinear partitions that decompose an n-gon into components that have no more than k vertices each, where k is given and $k < n$. Such a partition is called *k—partition*. A k-partition with a minimum number of components is called *minimum* k-partition. So far, the k-partition problem has been considered only for the cases $k = 4$ and $k = 6$. For the case $k = 4$, Lodi et al. [5] and Ohtsuki [6] give algorithms to find a minimum k-partition. For $k = 6$, the k-partition problem is closely related to the problem of stationing guards in rectilinear art galleries [2,4] where the interior of a rectilinear polygon shall be totally visible to a minimum number of stationary guards. An algorithm given by Edelsbrunner et al. [2] (originally for stationing guards) yields a minimum 6-partition if the input polygon is non-degenerate.

For two classes of rectilinear polygons we present algorithms to find a minimum k-partition for general k. For non-degenerate rectilinear polygons that are x-convex or y-convex the presented algorithm has time complexity $O(n)$. For degenerate polygons that are x-convex *and* y-convex an $O(n^4)$ algorithm is presented.

2. An Algorithm for Non-Degenerate Rectilinear Polygons

The algorithm *NONDEG* finds a minimum k-partition for a given non-degenerate rectilinear polygon that is x-convex or y-convex. W.l.o.g., *NONDEG* is described only for the case where the input polygon is x-convex.

Algorithm NONDEG

Input: An x-convex non-degenerate n-gon P, given by its vertex list.
Output: A minimum k-partition of P.

(1) Perform a horizontal plane sweep to find all horizontal non-degenerate cuts of P, sorted by y-coordinate.

(2) Partition P at each $\left\lfloor \frac{k-4}{2}+1 \right\rfloor$-th cut.

Theorem 1: *NONDEG* yields a minimum k-partition of an x-convex or y-convex non-degenerate polygon and has time complexity $O(n)$.

Proof: All components have no more than $\frac{k-4}{2}$ concave vertices, hence no more than k vertices. Thus, *NONDEG* yields a k-partition of P. P has $\frac{n-4}{2}$ concave vertices. Hence, *NONDEG* yields $\left\lfloor \frac{\frac{n-4}{2}}{\frac{k-4}{2}+1} \right\rfloor + 1 = \left\lceil \frac{n-2}{k-2} \right\rceil$ components. According to Günther [3], this is the minimum number of components achievable.

If P is x-convex, *NONDEG* requires a list of vertices of P that is sorted by y-coordinate. Due to the x-convexity of P this list can be obtained from the input vertex list in time $O(n)$ as follows. The vertex list of an x-convex polygon consists of at most three sublists that are alternately sorted by non-increasing and non-decreasing y-coordinate. Merging these sublists can be done in time $O(n)$ and yields a list of vertices that is sorted by y-coordinate. From there, steps (1) and (2) of *NONDEG* can clearly be performed in time $O(n)$ as well. A similar argument holds if P is y-convex. □

3. An Algorithm For Degenerate Rectilinear Polygons

In the degenerate case, things are more complicated. Section 3.1 presents the partitioning algorithm *DEG*. *DEG* has time complexity $O(n^4)$. Section 3.2 gives an example for the application of *DEG* to a rectilinear polygon. In section 3.3 it is proven that *DEG* yields a minimum k-partition of the input polygon.

3.1. The Algorithm DEG

A rectilinear polygon is called *magic* if it has $k + \lambda(k-2)$ vertices for some $\lambda \in \mathbf{N}$. *DEG* first partitions the input polygon along some of its degenerate cuts, such that all components are magic. Then it applies algorithm *NONDEG* to the single components. A more concise description of *DEG* follows; the subroutine *MAXSET*, that is called in step (2), is presented below.

Algorithm DEG

Input: An x-convex and y-convex n-gon P, given by its vertex list.

Output: A minimum k-partition of P.

(1) Perform horizontal and vertical plane sweep to find all degenerate cuts of P.

(2) Call subroutine *MAXSET* to find a maximum set of disjoint degenerate cuts of P whose elements partition P into magic components.

(3) Partition P along these cuts and apply *NONDEG* to each of the components.

Steps (1) and (3) of *DEG* clearly have time complexity $O(n)$. Before presenting the subroutine *MAXSET*, we need to introduce the following terms. Let e denote some edge of the input polygon P. A cut c partitions P into one subpolygon $P_e(c)$ that contains e, and one subpolygon $\overline{P_e(c)}$ that does not contain e. A degenerate cut dc is called *fit (with respect to e)* if for its subpolygon $\overline{P_e(dc)}$ either one of the following holds: $\overline{P_e(dc)}$ is magic, or there are degenerate cuts in $\overline{P_e(dc)}$ that partition $\overline{P_e(dc)}$ into magic components. The *order set* of a fit cut dc (w.r.t.e) is a maximum set of disjoint degenerate cuts of $\overline{P_e(dc)}$ whose elements partition $\overline{P_e(dc)}$ into magic components. The *order* of a degenerate cut dc (w.r.t.e) is 0 if dc is not fit (w.r.t.e); otherwise it is one plus the cardinality of its order set. To clarify these definitions, an example is given in figure 1.

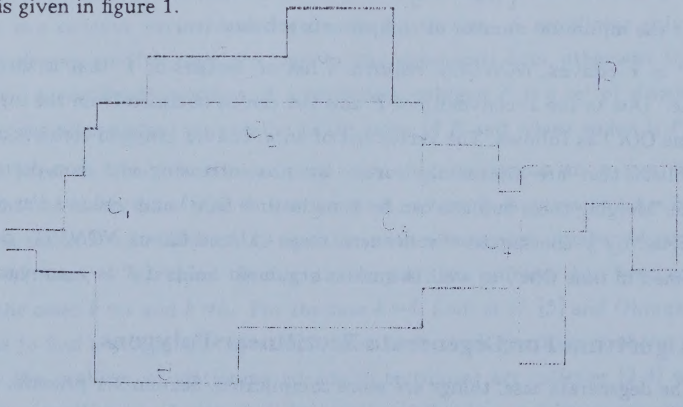


Figure 1: For $k=6$, c_1, c_2, c_3, and c_4 are fit cuts of polygon P. Their order sets are ϕ, ϕ, $\{c_4\}$, and ϕ, and their orders are one, one, two, and one.

MAXSET finds a maximum set of disjoint degenerate cuts of a rectilinear polygon P such that its elements partition P into magic components. In order to do so, *MAXSET* obtains the order and the order set of each degenerate cut of P. Finally, *MAXSET* combines these results to obtain a maximum set of disjoint degenerate cuts of P whose elements partition P into magic components. In order to perform this task efficiently, *MAXSET* employs a dynamic programming strategy [1], which utilizes the convexity of P. We only give a general outline of subroutine *MAXSET*. For a detailed presentation of *MAXSET* see [3].

Subroutine MAXSET

```
    Input: An $x$-convex and $y$-convex $n$-gon $P$, given by its vertex list, and a list of its degenerate cuts.
    Output: A maximum set of disjoint degenerate cuts of $P$ whose elements partition $P$ into magic components.
    (1) Let $e$ be the horizontal edge of $P$ with minimum $y$-coordinate.
    (2) Sort the degenerate cuts of $P$ into a list $L$ such that for each cut $c$ in $L$ the following holds: the degenerate cuts of $P$ that lie in polygon $\overline{P_e(c)}$ are predecessors of $c$ in list $L$.
    (3) Do for each cut $c$ in $L$:
        (3.1) Find all sets of disjoint fit cuts of polygon $\overline{P_e(c)}$ such that at most one cut in the set is horizontal and at most two cuts are vertical. Let $sc_k$ denote the cuts in the set.
        (3.2) Select all sets where
            (i) the polygons $\overline{P_e(sc_k)}$ are pairwise disjoint, and
            (ii) the polygon $\overline{P_e(c)} - \bigcup_k \overline{P_e(sc_k)}$ is magic.
        (3.3) If there are no such sets, assign 0 to the order of $c$ ($c$ is not fit w.r.t.$e$) and continue with the next cut in list $L$. Otherwise, for each set $S$ obtain the total order $TO_S$ of the cuts in $S$. Find the set $S_{\max}$ that maximizes $TO_S$. $TO_S+1$ is the order of cut $c$. The order set of $c$ is the union of the order sets of the cuts in $S_{\max}$ plus the cuts in $S_{\max}$ themselves.
    (4) Find all sets of disjoint fit cuts of polygon $P$ such that at most one cut in the set is horizontal and at most two cuts are vertical. Let $sc_k$ denote the cuts in the set.
    (5) Select all sets where
        (i) the polygons $\overline{P_e(sc_k)}$ are pairwise disjoint, and
        (ii) the polygon $P - \bigcup_k \overline{P_e(sc_k)}$ is magic.
```

(6) If there are no such sets, return the empty set and stop (*MAXSET*). Otherwise, for each set S obtain the total order TO_S of the cuts in S. Find the set $S_{\max}$ that maximizes TO_S. The solution set is the union of the order sets of the cuts in $S_{\max}$ plus the cuts in $S_{\max}$ themselves. This union set is a maximum set of disjoint degenerate cuts of P whose elements partition P into magic components. Return this set and stop (*MAXSET*).

Steps (1) and (2) can be performed in time $O(n)$. Steps (3.1) through (3.3) have time complexity $O(n^3)$, because there are $O(n^3)$ cut sets for a given cut c, and each set can be processed in time $O(1)$. Because there are $O(n)$ degenerate cuts in P, step (3) has time complexity $O(n^4)$. Steps (4) through (6) have time complexity $O(n^3)$. Hence, *MAXSEG* has time complexity $O(n^4)$. For the details of the complexity proof see [3].

3.2. An Example

An example for the application of *DEG* to a degenerate 80-gon with $\underline{k=8}$ is given in figures 2 and 3. Step (1) of *DEG* will yield the degenerate cuts of P:

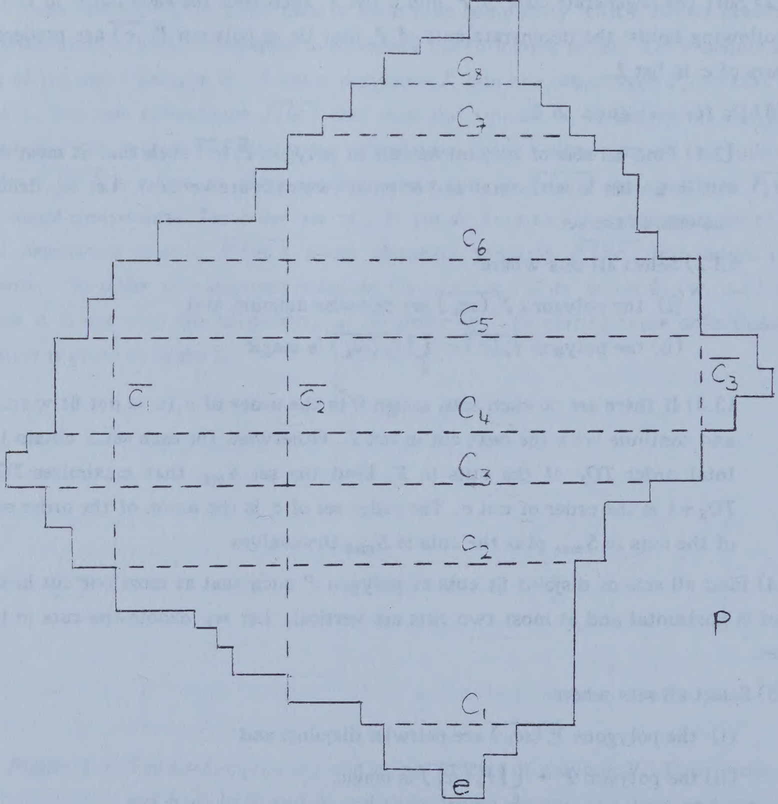


Figure 2: The 80-gon P and its degenerate cuts, marked by dashed lines.

Step (2) of subroutine *MAXSEG* will yield, for example, the following list L of degenerate cuts of P:

$$\overline{c}_1, \overline{c}_2, \overline{c}_3, c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1.$$

Step (3) of *MAXSEG*, applied to cut c_2, will find, for example, the following order set of c_2:

$$\{c_3, c_7, c_8, \overline{c_3}\}.$$

Hence, the order of c_2 is five. Note that the given order set is not the only possible one.

MAXSEG will return, for example, the following set of disjoint degenerate cuts of P:

$$\{c_1, c_7, c_8, \overline{c_1}, \overline{c_2}, \overline{c_3}\}.$$

Again, this is not the only possible solution. Another solution is:

$$\{c_1, c_2, c_3, c_7, c_8, \overline{c_3}\}.$$

Finally, step (3) of *DEG* partitions P along the degenerate cuts returned by *MAXSEG* and applies algorithm *NONDEG* to each of the components:

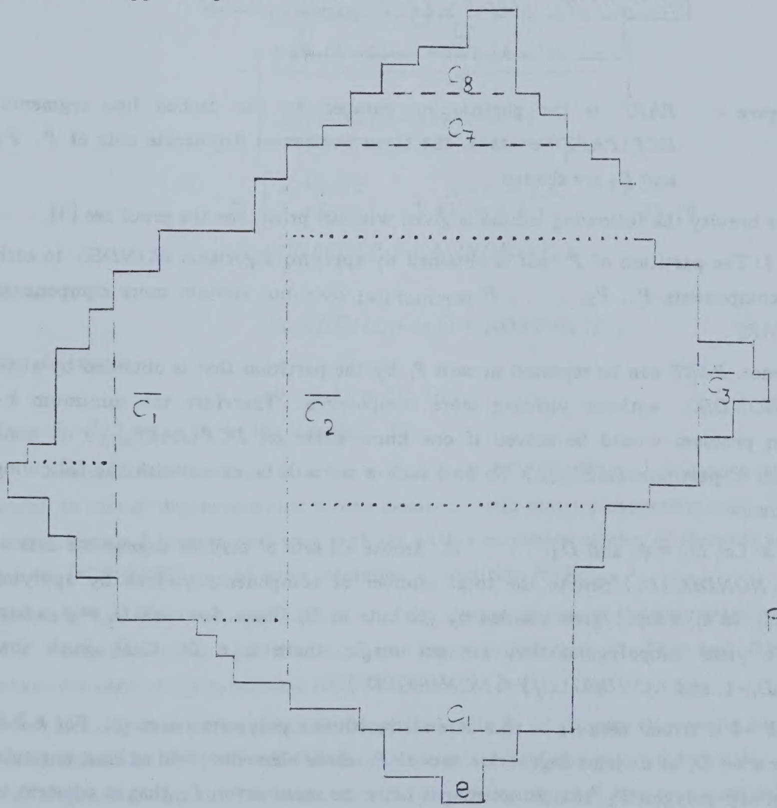


Figure 3: The dashed lines are the degenerate cuts returned by *MAXSET*. The dotted lines are the cuts yielded in step (3) of *DEG*.

3.3. Proof That DEG Yields A Minimum K-Partition

In this section it will be proven that *DEG* yields a *minimum* k-partition. Let *PART* be some arbitrary, but fixed k-partition of the x-convex and y-convex n-gon P. Let *SEG*(*PART*) denote the set of boundary segments of polygons in *PART*. Let *DCP*(*PART*) denote an arbitrary, but fixed maximum set of disjoint degenerate cuts of P that is in *SEG*(*PART*). Let $P_1, P_2, \ldots, P_{|DCP(PART)|+1}$ denote the components of P that are yielded by the degenerate cuts in *DCP*(*PART*). To clarify these definitions, an example is given in figure 4.

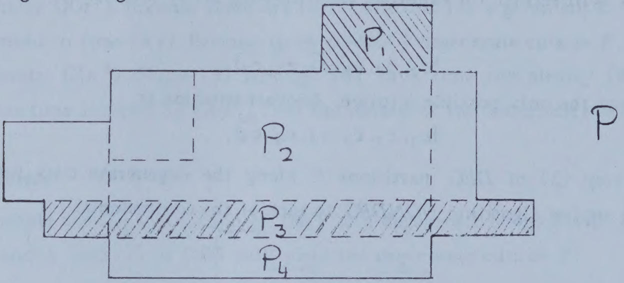


Figure 4: *PART* is the partitioning induced by the dashed line segments. *DCP*(*PART*) contains the three horizontai degenerate cuts of P. P_1 and P_3 are shaded.

For brevity the following lemma is given without proof; for the proof see [3].

Lemma 1: The partition of P that is obtained by applying algorithm *NONDEG* to each of the components $P_1, P_2, \ldots, P_{|DCP(PART)|+1}$ does not contain more components than *PART*.

Hence, *PART* can be replaced in each P_i by the partition that is obtained by algorithm *NONDEG*, withcut yielding more components. Therefore the minimum k-partition problem would be solved if one knew some set $DCP(PART_{MIN})$ of some minimum k-partition $PART_{MIN}$. To find such a set will be easier with the following two lemmas.

Lemma 2: Let $D_0 = \phi$, and $D_1, \ldots, D_r$ denote all sets of disjoint degenerate cuts of P, and $NONDEG(D_i)$ denote the total number of components yielded by applying *NONDEG* to all subpolygons yielded by the cuts in D_i. Then, for each $D_i \neq \phi$ whose elements yield subpolygons that are *not* magic, there is a $D_j \subset D_i$ such that $|D_j| = |D_i|-1$, and $NONDEG(D_j) \leqslant NONDEG(D_i)$.

Proof: $k=4$ is trivial because in this case all rectilinear polygons are magic. For $k \geqslant 6$, consider a set D_i of disjoint degenerate cuts of P whose elements yield at least one subpolygon, say polygon P_1, that is *not* magic. Let c be some cut in D_i that is adjacent to polygon P_1 and some other polygon P_2. Furthermore, let P_1 and P_2 have $k + \lambda_1(k-2) - X$ and $k + \lambda_2(k-2) - Y$ vertices, respectively, where

$X \in \{2,4, \ldots, k-4\}$, $Y \in \{0,2, \ldots, k-4\}$, $\lambda_1, \lambda_2 \in \mathbf{N}$. Note that P_1 cannot be magic. Let $NONDEG(Q)$ denote the number of components yielded by applying algorithm *NONDEG* to some rectilinear polygon Q. From the proof of theorem 1 it is known that for each n-gon Q

$$NONDEG(Q) = \left\lceil \frac{n-2}{k-2} \right\rceil .$$

Hence,

$$\begin{aligned} & NONDEG(P_1) + NONDEG(P_2) \\ &= \left\lceil \frac{k+\lambda_1(k-2)-X-2}{k-2} \right\rceil + \left\lceil \frac{k+\lambda_2(k-2)-Y-2}{k-2} \right\rceil \\ &= (\lambda_1+1)+(\lambda_2+1) \\ &= \lambda_1+\lambda_2+2 . \end{aligned}$$

On the other hand, $P_1 \bigcup P_2$ has $= (k+\lambda_1(k-2)-X)+(k+\lambda_2(k-2)-Y)$ vertices. Thus one obtains

$$\begin{aligned} & NONDEG(P_1 \bigcup P_2) \\ &= \left\lceil \frac{k+\lambda_1(k-2)-X+k+\lambda_2(k-2)-Y-2}{k-2} \right\rceil \\ &= \left\lceil \frac{(\lambda_1+\lambda_2+2)(k-2)+(2-X-Y)}{k-2} \right\rceil \\ &= \begin{cases} \lambda_1+\lambda_2+2 & X+Y = 2,4, \ldots, k-2 \\ \lambda_1+\lambda_2+1 & X+Y = k, k+2, \ldots, 2k-8 \end{cases} \\ &\leqslant NONDEG(P_1) + NONDEG(P_2) \end{aligned}$$

Hence, it is

$$NONDEG(D_i - \{c\}) \leqslant NONDEG(D_i).$$

and, of course

$$|D_i - \{c\}| = |D_i| - 1.$$

Thus $D_i - \{c\}$ is the D_j in the lemma. □

Lemma 2 implies that, in order to find a set $DCP(PART_{MIN})$, one can restrict its search to sets of degenerate cuts whose elements yield magic components only. The following lemma 3 implies that each such set with a *maximum number of elements* actually is a set $DCP(PART_{MIN})$ of some minimum k-partition $PART_{MIN}$.

Lemma 3: Let $DM_0 = \phi$, and $DM_1, \ldots, DM_q$ denote all sets of disjoint degenerate cuts of P whose cuts yield a partition of P into magic subpolygons P_i. Then each DM_i with maximum cardinality minimizes $NONDEG(DM_j)$ $(j = 0, \ldots, q)$.

Proof: Let DD denote the set of $D_0 = \phi$ plus all sets D_i of disjoint degenerate cuts of P. Furthermore, let $DDMAG \subset DD$ denote the set of $DM_0 = \phi$ plus all sets DM_j of disjoint degenerate cuts of P whose cuts yield a partitioning of P into magic subpolygons. Then it follows from lemma 2 that $DDMAG$ contains at least one set DM_{opt} such that

$$\text{for all } D_i \in DD: NONDEG(DM_{opt}) \leqslant NONDEG(D_i) .$$

Hence, what remains to prove is:

$$\text{for all } DM_j, DM_r \in DDMAG: \quad |DM_j| \geqslant |DM_r| \rightarrow NONDEG(DM_j) \leqslant NONDEG(DM_r) \qquad (*) .$$

Performing all cuts in DM_j ($DM_j \neq \phi$) yields a partitioning of P into $|DM_j|+1$ components P_i with $k+\lambda_1(k-2)$, $k+\lambda_2(k-2)$, . . . , $k+\lambda_{|DM_j|+1}(k-2)$ vertices, respectively ($\lambda_i \in \mathbf{N}$). It is

$$\sum_{i=1}^{|DM_j|+1} (k+\lambda_i(k-2)) = n$$

$$\rightarrow \quad (|DM_j|+1)k + (k-2) \sum_{i=1}^{|DM_j|+1} \lambda_i = n$$

$$\rightarrow \quad \sum_{i=1}^{|DM_j|+1} \lambda_i = \frac{n-(|DM_j|+1)k}{k-2} \qquad (**)$$

Hence,

$$\begin{aligned} &NONDEG(DM_j) \\ &= \sum_{i=1}^{|DM_j|+1} \left\lceil \frac{k+\lambda_i(k-2)-2}{k-2} \right\rceil \\ &= \sum_{i=1}^{|DM_j|+1} (\lambda_i+1) \\ &= \frac{n-(|DM_j|+1)k}{k-2} + (|DM_j|+1) \qquad \text{(by (**))} \\ &= \frac{n-2|DM_j|-2}{k-2} . \end{aligned}$$

Hence, one obtains

(i) for all DM_j, $DM_r \in DDMAG-\{DM_0\}$:

$$\begin{aligned} &|DM_j| \geqslant |DM_r| \\ &\rightarrow \frac{n-2|DM_j|-2}{k-2} \leqslant \frac{n-2|DM_r|-2}{k-2} \\ &\rightarrow NONDEG(DM_j) \leqslant NONDEG(DM_r) \end{aligned}$$

(ii) for all $DM_j \in DDMAG-\{DM_0\}$:

$$\begin{aligned} &NONDEG(DM_j) \\ &= \frac{n-2|DM_j|-2}{k-2} \\ &\leqslant \left\lceil \frac{n-2}{k-2} \right\rceil \\ &= NONDEG(DM_0) . \end{aligned}$$

From (i) and (ii), (*) follows immediately. □

From the above one obtains the following theorem.

Theorem 2: DEG yields a minimum k-partition of an x-convex and y-convex polygon in time complexity $O(n^4)$.

Proof: Lemmas 2 and 3 imply that a set D of disjoint degenerate cuts of a polygon P is a set $DCP(PART_{MIN})$ of some minimum k-partition $PART_{MIN}$ of P if

(i) all components yielded by the cuts in D are magic, and

(ii) D has maximum cardinality among all sets which fulfill condition (i).

The algorithm *DEG* finds such a set. The time complexity of *DEG* has been discussed above. From there the theorem follows. □

Acknowledgement

I would like to thank my advisor, Professor Th. Ottmann (U. of Karlsruhe, West Germany), for many clarifying discussions on earlier versions of this paper.

References

[1] Dreyfus, S., and Law, A., "The Art and Theory of Dynamic Programming", Academic Press, New York, 1977.

[2] Edelsbrunner, H., O'Rourke, J., and Welzl, E., "Stationing Guards in Rectilinear Art Galleries", Comp. Vision, Graphics, and Image Proc. 27, 1984, pp. 167-176.

[3] Günther, O., "Partitioning Algorithms for Rectilinear Polygons", Diplomarbeit, Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe, West Germany, 1984.

[4] Kahn, J., M. Klawe and D. Kleitman, "Traditional Galleries Require Fewer Watchmen", SIAM J. Alg. Disc. Meth., Vol. 4, No. 2, pp. 194-206, 1983.

[5] Lodi, E., Luccio, F., Mugnai, C., Pagli, L., and Lipski, W., "On Two-dimensional Data Organization II", Fundamenta Informaticae, Vol. 2, No. 3 (1979).

[6] Ohtsuki, T., "Minimum Dissection of Rectilinear Regions", Proc. IEEE Int. Symp. on Circuits and Systems, May 1982, pp. 1210-1213.

ON PARTITIONING RECTILINEAR POLYGONS INTO STAR-SHAPED POLYGONS

ROBIN LIU and SIMEON NTAFOS
Computer Science Program
University of Texas at Dallas
Richardson, TX 75080

ABSTRACT

In this paper we present an algorithm for finding optimum partitions of simple monotone rectilinear polygons into star-shaped polygons. The algorithm may introduce Steiner points and its time complexity is O(n), where n is the number of vertices in the polygon. We also present an approximation algorithm for partitioning simple rectilinear polygons into star-shaped polygons. The complexity of the approximation algorithm is O(nlogn) and the size of the partition obtained by the algorithm is no more than four times the size of an optimum solution.

1. INTRODUCTION.

Techniques to decompose polygons into simpler components have received widespread attention in pattern recognition, image analysis and computational geometry. The simpler components are usually convex, monotone, spiral or star-shaped polygons [10,16]. A polygon is simple if it contains no holes. A polygon is monotone if it contains two extreme vertices in a preferred direction which are connected by two polygonal chains that are monotonic in the preferred direction. Star-shaped polygons are defined in terms of visibility. A point x in polygon P is visible from a point y in P, if the open line segment joining x and y lies entirely inside P. A polygon P is star-shaped if there exists at least one point x inside P such that every point in P is visible from x. The set of all points with this property is the kernel of P.

An important class of polygons that occur in many applications (e.g., VLSI) is that of rectilinear polygons. A polygon is rectilinear if all its edges are either horizontal or vertical. In this paper we consider the problem of partitioning a simple rectilinear polygon into the minimum number of star-shaped polygons. The following results are known for this problem. Chvatal [4] and Fisk [6] considered the problem of stationing guards in an Art Gallery (Gallery Watchman Problem) and showed that any simple polygon can be partitioned into at most $\lfloor n/3 \rfloor$ star-shaped polygons without introducing Steiner points. Avis and Toussaint [2] gave an O(nlogn) algorithm to obtain partitions within this bound. Kahn, et. al. [8] showed that a simple rectilinear polygon can be partitioned into at most $\lfloor n/4 \rfloor$ star-shaped polygons without using Steiner points. Sack [15] gave an O(nlogn) algorithm to obtain such a partition (convex quadrilaterization). Related results on convex quadrilaterizations and the Gallery Watchman problem are given in [5, 12, 13]. Decompositions into triangles [7], rectangles [14], trapezoids [1] and convex polygons [3,9] also provide partitions into star-shaped polygons. All of these algorithms will not produce optimum partitions into star-shaped polygons in general. An algorithm that partitions simple polygons into the minimum number of star-shaped polygons was obtained by Keil [9]. This algorithm does not introduce Steiner points and its complexity is $O(n^5N^2 \log n)$, where n is the number of vertices and N is the number of notches (i.e., reflex vertices).

In the next section we present a linear time algorithm to partition a simple monotone rectilinear polygon into the minimum number of star shaped polygons when Steiner points are allowed. In section 3, we use this algorithm to obtain an approximation algorithm for partitioning non-monotone rectilinear polygons into star-shaped polygons.

2. DECOMPOSITION OF MONOTONE RECTILINEAR POLYGONS.

Consider a simple rectilinear polygon P that is monotone on the Y-axis. P consists of a unique topmost edge h_1, a unique bottom-most edge h_m and two chains, RC (right chain) and LC (left chain), such that the vertices in each of them project into a monotone sequence on the Y-axis. Assuming that P is given as a sequence of adjacent edges (or vertices), it is straightforward to obtain the list $H = \{h_1, h_2, \ldots, h_m\}$ of horizontal edges of P sorted from top to bottom. We associate the vertical edges of P with the edges in the list H by assigning each vertical edge to the horizontal edge that is adjacent and above it. Thus, h_1 has two vertical edges v_{1l} and v_{1r} associated with it, edge h_i is associated with v_i and h_m has no vertical edges associated with it. This results in the edge list $E = \{h_1, v_{1l}, v_{1r}, h_2, v_2, h_3, v_3, \ldots, h_m\}$. We also distinguish four types of edges according to the direction from which the edge bounds the interior of P. Thus we have top, bottom, left and right edges that bound the interior of P from the top, bottom, left and right respectively. Note that if e_k is a top edge in region R_i, then the kernel of R_i must lie below e_k. The other edge types restrict the kernel in analogous ways.

We obtain an optimum partition $R = \{R_1, \ldots, R_k\}$ of P by constructing the star shaped regions R_i one at a time proceeding from top to bottom. The method is essentially a greedy one of maximizing the number of contiguous edges in the current region. However, there are situations where the greedy approach does not lead to optimum solutions and appropriate modifications are needed. Also, in some cases, there may exist a number of choices involving a tradeoff between including more edges from one chain at the expense of including less edges from the opposite chain and an optimum choice can not be determined immediately. In such cases, all the viable solutions are kept and propagated until an optimal one among them can be determined.

The star shaped regions are defined by introducing k-1 cut edges $c_1, c_2, \ldots, c_{k-1}$, where c_i connects a point r_i in the right chain to a point l_i in the left chain (left, right cut-points) and forms the boundary between regions R_i and R_{i+1}. A cut edge c_i is called regular if r_i is to the right of l_i and is called a special cut edge otherwise. Similarly, a region and its kernel are called regular (special) if the upper cut edge of the region is regular (special). With respect to the remainder of the polygon, the cut edge is treated as an edge in the boundary of the polygon.

The first region is constructed by assigning to it the edges h_1 and v_{1l} , v_{1r}. Similarly, for region R_i, $i>1$, we start with the two topmost unassigned edges and the lower cut edge of the previous region. These three edges define a kernel which might not be bounded at the bottom. We augment the current region by adding to it edges in the order in which they appear in the edge list E and modifying the kernel accordingly. This process is terminated when addition of the next edge will result in an empty kernel. At this stage we have identified one of the cutpoints. Then we continue adding edges from the opposite chain in order to find the other cutpoint. Without loss of generality, let us assume that we find the left cutpoint first. Then we continue adding edges from the right chain as long as the following conditions hold:

C1: the new edge will not cause the kernel to become empty;
C2: the new edge does not advance below a bottom edge B in the left chain that will be assigned to the next region;
C3: the new edge does not advance to the left of a left edge L in the left chain that will be assigned to the next region.

We do not allow the cut edge to advance below B because that may produce non-optimum solutions. For example, consider the polygon shown in Figure 1. If we use c'_i as the cut edge, we need 3 regions while two regions are sufficient if we use c_i instead. We avoid this problem by checking the top vertex, v, of the last edge added from the right chain. If v is below the left cutpoint, we choose it as the right cutpoint. Otherwise, we introduce a Steiner point by placing the right cutpoint directly opposite the left cutpoint (see figure 2). A similar situation arises if the right cutpoint is allowed to advance past a left edge L that will be assigned to the next region (see figure 3). In this case we withdraw the right cutpoint to the previous vertex in the right chain (which is to the right of L). The algorithm to construct a regular region is shown in the following.

```
REGULAR-REGION
{this algorithm constructs a regular region; if E becomes
 empty the algorithm terminates}

a) Place the three topmost remaining edges in $R_i$ and use
   them to construct an initial kernel.
b) While addition of the next edge in E does not violate C1
   add the edge to $R_i$ and modify $K_i$ accordingly.
   Let e be the edge that violates C1.
   If $e \in LC$ then $l_i$ is the first vertex of e
              else $r_i$ is the first vertex of e.
c) {Assume that the left cutpoint is found in step b. The
    other case is similar}
   Remove the last edge from RC that was added
    to $R_i$ and insert it in E;
   While addition of the next edge in RC does not violate
   C1-C3, add the edge to $R_i$ and modify $K_i$ accordingly;
   Let e' be the edge that violates C1, C2 or C3;
```

```
If e' violates C1 or C3 then $r_i$ is the first vertex of e';
If e' violates C2 then
        if the first vertex of e' is below $l_i$,
            then $r_i$ is the first vertex of e'
            else $r_i$ is the Steiner point on e' that is
                   directly opposite of $l_i$;
Insert the cut edge $\{l_i, r_i\}$ in E.
```

Let c_i be the cut line created by REGULAR-REGION for R_i. If R_i is regular, then the corners in the lower side of its kernel uniquely determine the two cutpoints for c_i. That is, there is no cut line that satisfies conditions C1-C3 and contains more contiguous edges from either chain than c_i does. However, if the previous cut edge is special, it may cause the lower part of the kernel to be non-rectilinear (slanted). Then, the cutpoints can not always be determined uniquely from the kernel because the slanted part of the kernel may allow a tradeoff between including more edges from one chain but less edges from the opposite chain. In figure 4, three cut edges can be constructed by using points along this slanted edge where c_i is the edge created by REGULAR-REGION. Anyone of them can be the optimum choice depending on the shape of the remaining polygon. As shown in figure 4, determining which one of the cut edges is best may have to wait until we reach the last region in the polygon. Let C = $\{c_{i1}, c_{i2}, \ldots, c_{ij}\}$ be the various choices for the cut edge c_i that satisfy conditions C1-C3. We say that c_{ig} is a viable cut edge if there is no c_{ih} in C_i such that $h \neq g$ and neither of the cutpoints of c_{ih} occurs before a cutpoint of c_{ig} on the respective chains. We deal with special kernels by constructing all the viable cut edges. Assume that the special cut edge is as shown in Figure 4 (the other case is similar). We use REGULAR-REGION to obtain an initial cut edge. This edge has the property that it contains the most edges from the left chain and the least number of edges from the right chain among all the viable cut edges. Then we advance r_i and backup l_i using the kernel K_i to find the next viable edge and repeat until all the viable edges are found. The set of viable edges is kept in a stack. Any pair of edges in this set will intersect inside the polygon and each one represents a choice between including more edges from one chain and less edges from the opposite chain. Then, we remove a cut edge from the stack and construct the next region (using the method in REGULAR-REGION). We repeat this for each cut edge, each time comparing the resulting partitioning edge with the previous ones. If a pair of them does not intersect inside the polygon then we can safely select the lower of the two as a better choice. If they intersect inside the polygon, then both are viable choices and are added to the stack of viable cut edges for this region. Thus, by propagating all the viable choices we keep reducing the set of choices until eventually there is only one choice left or we reach the bottom of the polygon. If there is only one remaining choice, then this determines a set of proper choices for the previous regions. If we reach the bottom of the polygon and more than one viable choice remains, then we select the first one of them and make the corresponding selections in the previous regions. The algorithm (SPECIAL-REGION) is given in the following.

```
SPECIAL-REGION.
 {This algorithm handles special regions in which a number of
  choices for cut edge need to be propagated.  If the list E
  becomes empty the algorithm terminates}
 a) {Create set of viable cut edges for $R_i$, the first
     special region with multiple choices for $c_i$}
    Initialize stacks S1, S2 to empty;
    Use REGULAR-REGION to construct $c_{i1}$; PUSH($c_{i1}$,S1);
    Let T be the last top edge in the RC of $R_{i1}$;
    While T remains in $R_i$ and C1-C3 are not violated
      Construct the next viable choice for $c_i$ and PUSH it onto
      S1; {use $K_i$; the right cutpoint moves down RC while
          the left moves up LC}
    Save S1 and let j = i;

 b) {Propagate the viable choices}
    While |S1| > 1
     P1 <-- POP (S1); {P1 has the highest left and lowest
                       right cutpoint}
     j = j+1;
     Construct c' = $c_{j1}$ using REGULAR-REGION;
     If c' is special, then STOP; {$c_{j1}$ is the best choice}
     If the right chain in $R_{j1}$ includes a top edge
        then P2<--POP(S1);
             Construct $c_{j2}$ (corresponding to P2) using
               REGULAR-REGION;
             Select the lower of $c_{j1}$, $c_{j2}$ and STOP;
        else Find the bottom, left and top boundaries of the
             kernels corresponding to the cut edges in S1.
             PUSH(c',S2);
             While S1 is not empty and conditions C1-C3 are
               not violated
               begin
               P2<--POP(S1).
               Use the kernel corresponding to P2 to create
               c''; (the next choice for $c_j$)
               If c'' is below c'
                 then POP(S2); PUSH(c'',S2);
               If c'' and c' intersect, then PUSH(c'',S2);
               If c'' is special, then STOP.
               c' = c'';
               end
             Save S2, let S1 = S2 and set S2 to empty;

c) Use the unique cut edge $c_j$ to find the cut edges $c_i$, $c_{i+1}$,
   ...., $c_{j-1}$.
```

To partition a monotone rectilinear polygon into the minimum number of star-shaped polygons, we start with REGULAR-REGION and use REGULAR-REGION and SPECIAL-REGION repeatedly (depending on the last cut edge that was constructed) until all the edges are exhausted.

<u>Theorem 1</u>. Let R = $\{R_1, R_2, \ldots, R_n\}$ be the solution obtained by the algorithm and let Q = $\{Q_1, Q_2, \ldots, Q_{n'}\}$ be any solution. Then, $n \leq n'$.

<u>Proof</u>: The proof (see [11]) is by induction on the size of the partition obtained by the algorithm. We show that in

any solution there are at least i kernels above c_i and the corresponding regions can not include some edge below c_i.

The complexity of the algorithm is O(n), where n is the number of vertices in the polygon. This follows from the following analysis. The edge list can be constructed in linear time by merging the two monotone chains. Each kernel can have at most 5 sides and updating it every time an edge is added takes constant time (per edge). The construction described in REGULAR-REGION looks at each edge in the current region once except when a Steiner point is introduced. A vertical edge which has m Steiner points on it may be handled 2(m+1) times. Since this vertical edge is split among m+1 regions, we can distribute the costs to these m+1 regions and the cost per region is constant. REGULAR-REGION may consider some edges that will belong to the next region in order to find the first bottom and the first vertical edge needed to check conditions C2, C3. By looking for these edges in one chain as the construction proceeds on the opposite chain we guarantee that we never look past the next region. This cost can be attributed to the edges in the next region.

In SPECIAL-REGION we need to keep track and propagate a number of possible solutions. The initial choice for a cut line in a special region is obtained using REGULAR-REGION, so it takes linear time. The remaining choices for cut edge are constructed by considering new edges in one of the chains and reconsidering edges in the opposite chain. We can assign the cost of each cut line (and the corresponding kernel) to the edges that give rise to it. The number of times we consider an edge in the first special region is at most 3 (one time for information needed by the previous region, one time to construct the first cut edge, one time when we back up in one chain in order to construct the remaining choices for cut edge). Note that there is a natural order for the cut edges and the polygon edges that give rise to the corresponding kernels so we can find all the viable cut edges by backing up only one time. If propagation takes place, note that the number of viable cut edges per region can not increase. This follows from the fact that only special edges result in multiple viable cuts and if the first viable choice along the propagation is a special cut edge then propagation stops there. The difference between the initial special region and regions during propagation is only in that the kernels of the latter are slightly more difficult to construct. The left and bottom (for the case we consider in the algorithm) boundaries can be determined as we construct the initial cut edge. Finding the top boundaries requires that we back up in one of the chains but again the boundaries are found in order and we only need to back up one time. When the first viable choice constructs a region which includes a top edge, we use REGULAR-REGION to construct a region for the second choice and may need to use REGULAR-REGION twice for the last propagation region.

Thus the time complexity of the algorithm is O(n), where n is the number of vertices in the polygon.

3. DECOMPOSITION OF RECTILINEAR POLYGONS.

In this section we use the algorithm described in the previous section to obtain an approximate solution to the problem of partitioning a simple rectilinear polygon into star-shaped polygons. Given a simple polygon P, we first partition P into monotone rectilinear polygons and then we use the algorithm to obtain an optimum solution for each monotone rectilinear polygon in the partition. The approximate solution is obtained by taking the union of the resulting partial solutions.

A top edge is called a top peak if both vertical edges adjacent to it are above it. Bottom, left and right peaks are defined similarly. Let Y be the number of top and bottom peaks in P and X be the number of left and right peaks in P. If $Y > X$, we rotate the polygon by ninety degrees. Since there must be a cutpoint in each peak, we have:

Lemma 1. Let n be the number of top and bottom peaks in P and let m be the number of components in an optimum partition of P into star-shaped polygons. Then, $n < m$.

Let T be a top edge and B a bottom edge in P. Let PT (PB) be the projections of T (B) onto the x-axis. We say that T (B) is related to B (T) if PT and PB overlap.

We obtain an approximate solution by the following algorithm.

1) Partition P into monotone rectilinear polygons M_1, M_2, $\dots, M_k$.
2) For each M_i, $1 \leq i \leq k$, use the algorithm of Section 2 to obtain an optimum partition, A_i, into star-shaped polygons.
3) The partition of P is $P = \bigcup_i A_i$.

Step 1 can be carried out in O(nlogn) by sorting the horizontal edges, inserting the top (bottom) peaks into a B-tree and for each bottom (top) edge searching the tree for peaks related to it and introducing appropriate cut edges.

Let $c_1, c_2, \dots, c_{k-1}$ be the cut edges and $M_1, M_2, \dots, M_k$ be the monotone rectilinear regions obtained using the algorithm above. Consider a partition Q of P into star-shaped polygons. We can adjust Q so that none of the cut edges c_i intersects a star-shaped region in Q more than once. Consider the partition Q' obtained from Q by introducing the cut edges c_i, $1 \leq i \leq k-1$. There are four types of regions in Q'. Let T_i be the regions in Q' that also appear in Q and are within M_i (i.e., the regions in Q that lie completely in M_i). Of the remaining regions in Q', let B_i, F_i be those that lie in M_i and border only a right or left cut edge of M_i respectively. Finally let P_i be the regions in Q' that lie in M_i and border both a left and right cut edges of M_i. Let R_i be the solution obtained by the algorithm described in the previous section for the monotone polygon M_i. Then the optimality of the algorithm implies that:

$$\sum_i (T_i+B_i+F_i+P_i) \geq \sum_i R_i \quad \text{or} \quad \sum_j (T_j+B_j+F_j+P_j) \geq \sum_i R_i \qquad (1)$$

where T_j, B_j, F_j, P_j are those regions in Q' that form the j region in Q.

Theorem 2. Let Q be any partition of P into star-shaped polygons and let $P = U_i \, A_i$ be the partition obtained by the approximation algorithm. Then $|P| < 4|Q|$.

Proof: First we assign some of the regions in Q' to corresponding regions in Q. If $|T_j| = 1$, we assign it to Q_j. If T_j is empty, then we assign to Q two regions in $B_j U F_j U P_j$ that contain the edges that form the top and bottom boundaries of the kernel of Q_j (if both of those edges are in the same region, we assign an arbitrary second region to Q_j). The number of regions in Q' thus assigned to Q is at most $2|Q|$. Of the remaining regions in Q', some are merged into bigger regions and others are assigned to the cut edges c_i, $1 \leq i \leq k-1$. Consider the unassigned regions in Q' that lie in M_i. If M_i lies between two cut edges, (Figure 5), then the unassigned regions can be merged into just two regions, one at the top and another at the bottom. If the border of M_i is formed by more than two cut edges, then the unassigned regions in M_i can be merged into at most $2(C_i-1)$ regions, where C_i is the number of cut edges in the border of M_i. Note that relation (1) holds even after merging. If we move from the left and right borders of the polygon towards the middle and assign up to two non mergeable regions in each M_i to the cut edge(s) that was crossed in entering M_i, then all the non mergeable regions can be assigned to the cut edges with at most two assignments per cut edge. Since $|Q|$ is greater than the number of cut edges c_i (by lemma 1), it follows that $2|Q|$ is greater than the number of regions in Q' that are assigned to cut edges.

We have accounted for all the regions in Q' by a) assigning some to regions in Q (total is less than $2|Q|$), b) merging regions within each M into at most two regions per cut edge (relation (1) still holds) and c) assigning the merged regions to the cut edges (total is less than $2|Q|$). Then, from relation (1), it follows that $|R| < 4|Q|$.

4. CONCLUSIONS.

We have presented a linear time algorithm to partition a simple monotone rectilinear polygon into star shaped polygons. This algorithm can be used to obtain partitions of simple rectilinear polygons whose size is no more than four times the optimum. We conjecture that the solutions obtained are within a factor of three of the optimum solution. Further improvements in the solution may be possible by extending the algorithm of section 2 to 'nearly' rectilinear monotone or monotone polygons.

REFERENCES.

1. T. Asano and T. Asano, "Minimum partition of polygonal regions into trapezoids," in Proc. 24th IEEE Annu. Symp. Foundations Comput. Sci., Nov. 1983, pp. 233-241.
2. D. Avis and G.T. Toussaint, "An efficient algorithm for decomposing a polygon into star-shaped polygons," Pattern Recognition, Vol. 13, No. 6, 1981, pp. 395-398.
3. B. Chazelle and D. Dobkin, "Decomposing a polygon into its convex parts," in Proc. 11th Annual ACM Symp. on Theory of Computing, 1979, pp. 34-48.

4. V. Chvatal, "A combinatorial theorem in plane geometry," J. of Combinatorial Theory B, Vol. 18, 1975, pp. 39-41.
5. H. Edelsbrunner, J. O'Rourke and E. Welzl, "Stationing guards in rectilinear art galleries," Computer Vision, Graphics and Image Processing, Vol.27, 1984, pp.167-176.
6. S. Fisk, "A short proof of Chvatal's watchman theorem," J. of Combinatorial Theory B, Vol. 24, 1978, pp. 374.
7. M.R. Garey, D.S. Johnson, F.P. Preparata and R.E. Tarjan, "Triangulating a simple polygon," Information Processing Letters, Vol. 7, No. 4, 1978, pp. 175-179.
8. J. Kahn, M. Klawe and D. Kleitman, "Traditional galleries require fewer watchman," SIAM J. Algebraic and Discrete Methods, Vol. 4, No. 2, 1983, pp. 194-206.
9. J.M. Keil,"Decomposing polygons into simpler components," Ph.D. dissertation, Tech. report #163, Dept. of Computer Science, Univ. of Toronto, 1983.
10. D.T. Lee and F.P. Preparata, "Computational Geometry - a Survey," IEEE Trans. on Computers, Vol. C-33, No. 12, Dec. 1984, pp. 1072-1101.
11. R. Liu and S. Ntafos, "On Partitioning Rectilinear Polygons into Star-Shaped Polygons," Tech. report # 216, Computer Science Program, Univ. of Texas at Dallas, 1985.
12. Anna Lubiw, "Decomposing polygonal regions into convex quadrilaterals," in Proc. of the Symp. on Computational Geometry, 1985, pp. 97-106.
13. J. O'Rourke, "An alternate proof the rectilinear art gallery theorem," Journal of Geometry, Vol. 21, 1983, pp. 118-130.
14. L. Pagli, E. Lodi, F. Luccio, C. Mugnai and W. Lipski, "On two dimensional data organization 2," Fundamenta Informaticae, Vol. 2, No. 3, 1979.
15. J.R. Sack, "An O(nlogn) algorithm for decomposing simple rectilinear polygons into convex quadrilaterals," in Proc. 20th Allerton Conf., 1982, pp. 64-74.
16. G.T. Toussaint, "Pattern recognition and geometrical complexity," in Proc. Fifth International Conf. on Pattern Recognition, Dec. 1980, pp. 1324-1347.

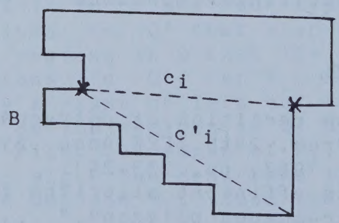


Figure 1

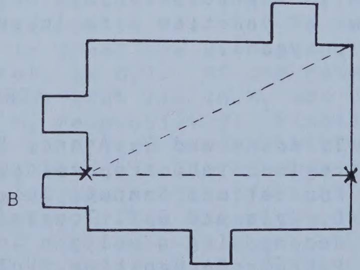


Figure 2

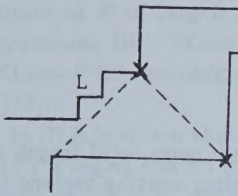


Figure 3

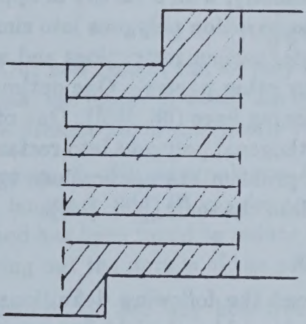


Figure 5

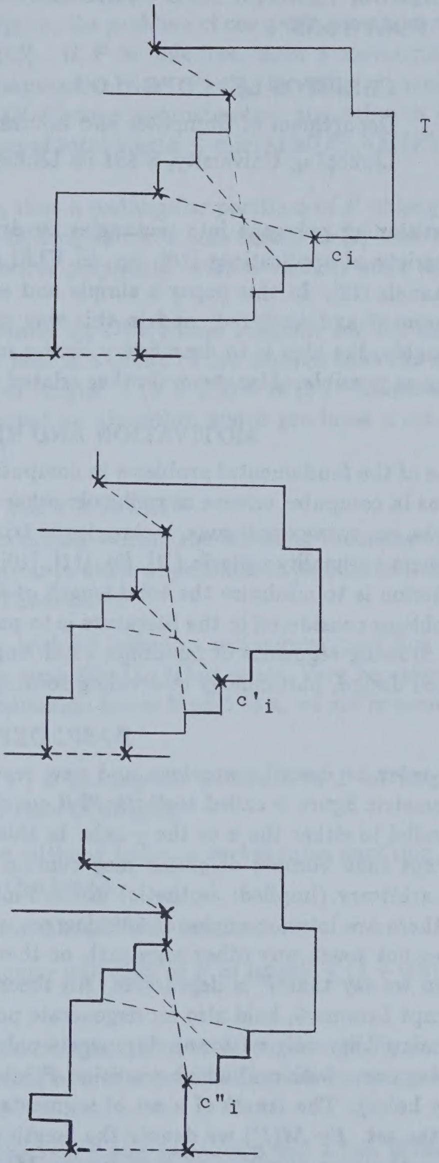


Figure 4

MINIMUM LENGTH AND "THICKEST-FIRST" RECTANGULAR PARTITIONS OF POLYGONS

CHRISTOS LEVCOPOULOS
Department of Computer and Information Science
Linköping University, S-581 83 Linköping, Sweden

Partitioning polygons into rectangles by drawing edges of minimum total length has a variety of applications [10], eg. in VLSI design when dividing routing regions into channels [12]. In this paper a simple and efficient greedy approximation algorithm is presented and analyzed, and in this way previous results are improved significantly. Roughly, the idea is to draw every time a maximal rectangle whose shortest side is as long as possible. Also, several other related results are announced.

MOTIVATION AND RELATED PROBLEMS

One of the fundamental problems in computational geometry, with a variety of applications in computer science as well as in other fields, is to partition polygons into simpler parts, eg. convex polygons, rectangles or triangles, under various restrictions and using various optimality criteria ([1], [9], [11], [10], and many other papers). One optimality criterion is to minimize the total length of the partitioning lines ([9], [10]). One of the problems considered in the literature is to partition orthogonal polygons into rectangles by drawing segments of minimum total length. This problem has applications eg. in VLSI design, particularly in dividing routing regions into channels ([12], [10]).

BASIC DEFINITIONS

In order to describe previous and new results we need the following definitions. A geometric figure is called *isothetic* iff it consists of points and/or straight line segments parallel to either the x or the y axis. In this paper we deal only with isothetic objects, except that Voronoi diagrams may contain non-isothetic segments. We denote by P an arbitrary (implied: isothetic) not self-intersecting polygon, with or without holes. If there are interior angles of 360 degrees within P (eg. one segment within P which does not touch any other segment), or there is some hole within P which is a point, then we say that P is *degenerate*. All theorems and lemmas mentioned in this paper, except Lemma 6, hold also for degenerate polygons, but within this paper we shall deal for simplicity only with non-degenerate polygons. A *rectangular partition* of P is a set of segments within P which partition P into rectangles (the rectangles do not contain any holes). The *length* of a set of segments is the sum of the lengths of the segments in the set. By $M(P)$ we denote the length of any shortest rectangular partition of P. A rectangular partition of P of length $M(P)$ is called a *minimum length rectangular partition* of P. Moreover, n is the number of vertices of P and $p(P)$ is the length of the perimeter of P (including the perimeter of its holes, if it has any).

We say that an (isothetic) polygon P without holes is a *histogram* (sometimes called *Manhattan polygon* in the literature), iff it has a distinguished edge (arbitrarily chosen), called the *basis* of P, such that the length of the basis of P is equal to the total length of all other edges of P which are parallel to the basis.

OVERVIEW OF PREVIOUS RESULTS

When polygonal holes are allowed in the polygon, the problem of computing a minimum length rectangular partition is NP-hard [10]. If P is hole-free, then a rectangular partition of P of length $M(P)$ can be computed in time $O(n^4)$, by using dynamic programming [10]. When P has holes, an $O(n^4)$-time approximation algorithm in [8] partitions P into rectangles by drawing edges of total length $\leq \min(41M(P), 4M(P) + p(P)/2)$.

In [7] it is shown that if P has no holes, then a rectangular partition of P of length $O(p(P) \times \log n)$ can be computed in time $O(n \log n)$. It is also shown in [7] that for all even numbers $n \geq 6$ there exists a hole-free polygon P with n vertices such that $M(P) = \Omega(p(P) \times \log n)$.

Gonzalez and Zheng [4] presented recently an $O(n^2)$-time heuristic for the case when the polygon is a rectangle and every hole is a point. They showed that the set of segments specified by their heuristic is of length $\leq (3 + \sqrt{3}) \times M(P)$. Moreover, by using ideas similar as in [8], they presented an algorithm which produces a set of segments of length $\leq 3M(P)$ in time $O(n^4)$.

OVERVIEW OF NEW RESULTS

Our new results can be very briefly summarized within the following theorems and lemmas, the proofs of which are going to appear in other papers and in [6] (the contents of the present paper are mostly related to Theorem 1).

Theorem 1. For any polygon P (with or without holes), a rectangular partition of P of length $< 5M(P)$ can be computed in time $O(n^2)$. (Comment: Very recently a method has been found to reduce the approximation factor from 5 to 4, we are presently working out the details of the proof).

Theorem 2. For any hole-free polygon P, a rectangular partition of P of length $< p(P) + 2.42 \times M(P)$ can be computed in time $O(n \log n)$.

Theorem 3. For any polygon P, with or without holes, a rectangular partition of length $O(M(P))$ can be computed in time $O(n \log n)$.

In particular, Theorem 2 is derived from the following two lemmas:

Lemma 1. For any histogram P, a rectangular partition of P of length $\leq (1 + \sqrt{2}) \times M(P)$ can be produced in time $O(n \log n)$.

Lemma 2. For any hole-free polygon P, a set of segments of total length $< p(P)$, partitioning P into histograms, without introducing new concave vertices, can be computed in time $O(n \log n)$.

In order to prove Theorem 3 we use Theorem 2 and Lemmas 3 and 4 (see below). In particular, in order to obtain the result claimed in Theorem 3, first we apply the transformation mentioned in Lemma 3 to the polygon, then the resulting polygon is transformed according to Lemma 4, and the result is processed according to Theorem 2.

Lemma 3. For any polygon P, it is possible to cut off P a linear number of rectangles, in time $O(n \log n)$ by cutting along segments of total length $O(M(P))$, such that if P' is the resulting polygon, then $M(P') + p(P') = O(M(P))$.

Lemma 4. Let P be any polygon with holes. In time $O(n \log n)$ it is possible to connect the holes to the boundary, by drawing a minimum length rectilinear spanning tree (without Steiner points) [5], of length $\leq 2M(P)$), using the generalized Voronoi diagram where the convex distance function is the isothetic square turned 45 degrees ([2], [3]). By a slight modification of a proof in [8] it can be shown that the length of the tree is $\leq 2M(P)$, and that, if P' is the resulting hole-free polygon, then $M(P') \leq 2M(P) + p(P)/2$.

RESULTS WHICH ARE PROVED IN THIS PAPER

In the continuation of this paper we concentrate mainly on the result claimed in Theorem 1. In the following section the algorithm is presented which achieves this result. The algorithm produces so-called "thickest-first" partitions in time $O(n^2)$. Although thickest-first partitions have length almost $5M(P)$ in worst case, the algorithm has the advantage of being simple and elegant, and we have found that in most cases the produced result is very close to the optimum. Therefore, although there are algorithms with better worst-case performance, the "thickest-first" algorithm may be best for practical use.

Because of space considerations, no proof of the approximation factor 5, claimed in Theorem 1, is given in this paper. Instead, the main ideas are illustrated by proving that thickest-first partitions have length less than $8M(P)$ when the polygons are non-degenerate. For the case when degeneracies occur, the same result can be proved analogously, but a few additional cases have to be considered. A full proof of a strengthened version of Theorem 1 (with better approximation factor) will appear in [6].

Finally, the special case when the input polygon is a histogram is treated in more detail (cf. Theorem 2 and Lemmas 1 and 2). It is shown how the idea of thickest-first partitions can be trimmed for this special case in order to improve the approximation factor to $1 + \sqrt{2}$.

"THICKEST-FIRST" PARTITIONS AND HOW TO COMPUTE THEM

In this section we define what is a thickest-first partition and we show how to compute such partitions in $O(n^2)$ time.

To go on we need the following definitions. We say that a rectangle R is *thicker* than a rectangle R' iff the shortest side of R is longer than the shortest side of R'. Let R be a rectangle within P. The rectangle is called *maximal* (with respect to P) iff it is not properly included in any other rectangle lying within P. We say that R is a *thickest* rectangle within P iff there is no thicker rectangle lying within P (recall that we consider only isothetic objects). Let S be a set of segments. We say that S is a *thickest-first* rectangular partition of P iff it consists of disjoint segments, lying within P and not overlapping with the boundary of P (overlappings are allowed only at endpoints), and the rectangles into which P is partitioned by S can be drawn by the following recursive procedure: If P is a rectangle, then do not partition it. Otherwise let R be any thickest maximal rectangle within P. Draw R, and continue recursively, in the same way, with the subpolygons of P in the partition induced by R.

A simple method to compute a thickest-first partition in time $O(n^2)$ is to construct a generalized Voronoi diagram, where the convex distance function is the isothetic square ([2], [3]). This diagram can be constructed in time $O(n \log n)$ [3]. Given such a diagram, a maximal thickest rectangle within P can be found in linear time, by scanning the vertices and the edges of the diagram. The new Voronoi diagram for the figure resulting by drawing this rectangle can be computed in linear time, by updating the old diagram. The algorithm continues in this way with the resulting figure until it is totally partitioned into rectangles. It is easily seen that the total number of rectangles produced is not greater than the number of concave vertices of the polygon, and thus the total time needed by the algorithm is $O(n^2)$. The details of the algorithm are easy to derive and are therefore omitted.

THICKEST-FIRST PARTITIONS ARE NOT MUCH LONGER THAN MINIMUM LENGTH RECTANGULAR PARTITIONS

Within this section it is shown that for any non-degenerate polygon P, every thickest-first partition of P is of length $< 8M(P)$. In order to enhance the intuition of the reader, we start by stating the following result without proof (a strengthened version of this result is shown later, see Lemma 6).

Fact 1: Let P be an arbitrary polygon with ≥ 6 vertices, and let R be any maximal thickest rectangle within P. Let l_R be the length of the part of the perimeter of R which is disjoint from the boundary of P. Let S be an arbitrary rectangular partition of P. It holds that the total length of the intersection of the segments in S with the closed region bounded by R is greater than $l_R/3$ (if degeneracies occur, then the fact is true if the term $l_R/3$ is replaced by $l_R/4$).

By using the above fact, it can be proved that if no new concave vertices are created during the process of drawing thickest maximal rectangles, then the length of the thickest-first partition is less than three times the optimum. For example, it is always so when the polygon is a histogram (see last section).

In order to handle the case when new concave vertices are created, a pseudo-algorithm is used, called *PROV8*, (not intended to be implemented, it only helps the proof), in order to associate segments in thickest first partitions to segments in optimal partitions. Let S be a rectangular partition of P of length $M(P)$. It is helpful to assume that the segments of S are drawn with blue color initially, and that the boundary of P is black. The procedure *PROV8* draws rectangles using black color, and it erases blue segments. It also marks concave vertices with lengths, ie. real numbers and some length unit. The purpose, roughly, is to indicate the length of the segments for which these concave vertices are able to "pay". In the beginning all vertices are unmarked. The procedure works as follows. The basic step is described in a more general way, which allows recursive application to subpolygons of P. Let R be any thickest, maximal rectangle within P. If $R = P$ then exit. Otherwise, let l_{blue} be the total length of all blue (parts of) segments lying in the closed region bounded by R. Let l_{vert} be the sum of all lengths with which the marked concave vertices of P lying on the boundary of R are marked. Also, let l_R be the total length of the part of the boundary of R which does not lie on the perimeter of P. Let l_{rem} be equal to $8 \times l_{blue} + l_{vert} - l_R$. Finally, let i be the number of corners of R which do not lie neither on a black nor on a blue segment. If $i \geq 1$, then mark every one of these i corners with the number l_{rem}/i. Unmark all marked vertices of P lying on the perimeter of R. Finally, erase all blue (parts of) segments which lie in the closed region bounded by R, draw (with black color) the part of the boundary of R which is not black already, and continue recursively in the same way with the subpolygons of P in the partition of P induced by the drawn segments.

One important lemma is the following:

Lemma 5: For every step of *PROV8* above, it holds that $l_{rem} > i \times 2s$, where s is the length of the shortest side of R. (Comment: This lemma holds also if degeneracies occur, but a few additional cases have to be considered in order to prove this).

By using this lemma it is not difficult to obtain the following conclusion:

Conclusion 1: For any non-degenerate polygon P, the length of every thickest first partition of P is $< 8M(P)$.

To prove Lemma 5, we have to prove first a stronger version of Fact 1, namely the following lemma:

Lemma 6: We assume the above context. Let m be the number of marked vertices lying on R. Then it holds that $l_R < 3 \times l_{blue} + m \times s$ (the constant 3 would have to be replaced by 4 if degeneracies occur).

Proof of Lemma 6.

Let us consider first the easy case when every isothetic line passing through R intersects (or touches) some blue segment within the closed region bounded by R. In this case it is easily seen that the total length of the vertical, respectively horizontal, blue segments within this region is not less than the length of the vertical, respectively horizontal, side of R. Thus, the total length of all blue segments in the region is not less than half of the perimeter of R, and the lemma trivially follows, independently of the number of marked vertices.

So it remains to consider the case when there a exists a line l, w.l.o.g. horizontal, such that l passes through R, and it does not intersect (neither touch) any blue segment within the region.

For easier reference we color some parts of the perimeter green or red. Let us color green the parts of the perimeter of R which do not overlap with the boundary of P neither with blue segments. After this, let us change to red all green segments which do not touch any marked vertex. Let l'_{blue} be the total length of the blue segments segments lying in the open region bounded by R, ie. the closed region excluding the perimeter of R. To prove the lemma it suffices to prove that the total length of the green and red segments together is less than $3 \times l'_{blue} + m \times s$.

A crucial observation is the following:

Observation 1 : No green nor any red segment is longer than s.

Proof : If this would not hold, then it is easily seen that it would be possible to place within P a rectangle thicker than R, namely one overlapping with R and being crossed by such a long green or red segment. (Comment: This would not be necessarily true if degeneracies were allowed). [Q.E.D.]

Thus the total length of the green segments is not greater than $m \times s$. It remains to show that the total length of the red segments (if there are any) is less than $3 \times l'_{blue}$. This follows immediately from Propositions 1 and 2 stated below.

Proposition 1 : The total length of the horizontal red segments, if there are any, is less than l'_{blue}.

Proposition 2 : The total length of the vertical red segments (if there are any) is not greater than $2 \times l'_{blue}$.

To prove the propositions, it is important to establish first the following fact.

Observation 2 : Each red segment touches some blue segment, such that this blue segment overlaps with the open region bounded by R.

Proof : Each unmarked concave vertex of P must touch some blue segment which cancels it, and the same holds for endpoints of blue segments which are not on the boundary of P, and since the red segment does not overlap with any blue, the cancelling segment has to overlap with the open region bounded by R. [Q.E.D.]

Proof of Proposition 1 : To see that the proposition is true, let us hypothesize that there is at least one red segment above the horizontal line l. Then, according to Observation 2, there must exist blue segments above l in the open region bounded by R. Since no blue segment touches nor crosses l, the only way by which the concavities occurring at the endpoints of blue segments can be cancelled in the blue partition is a blue path above l from the left side of R to its right side. This blue path is not shorter than the horizontal side of R, and therefore its length is greater than the total length of the red horizontal segments above l. Analogously we conclude that the total length of the horizontal red segments below l, if there are any, is less than the total length of (parts of) blue segments in the part of the open region bounded by R which is below l, and the proposition easily follows. [Q.E.D.]

Proof of Proposition 2 : To show Proposition 2, it is enough to prove, w.l.o.g., that the total length of the vertical red segments on the left side of R is not greater than l'_{blue}. The corresponding proof for the right-hand vertical red segments is analogous.

With every left red vertical segment, we associate uniquely (parts of) blue segments of total length no less than the length of the red segment as follows. For each such red segment a rectangle is defined, say R', such that the basis of R' is the red segment, R' lies within R and the length of its horizontal side is equal to the length of the horizontal side of R. Clearly, no two rectangles defined in this way touch each other, so it is enough to show that the length of the red segment is not greater than the length of the (parts of) blue segments which lie in the closed region bounded by R' but not on the perimeter of R. This is proved by observing first, according to Observation 2, that at least one blue segment lies within this region. Thus, since no concavities are build by the blue segments, there exists in the defined region a blue path connecting either the two horizontal, or the two vertical sides of R'. Since the length of the red segment is not greater than s, it follows that the horizontal sides of R' are not shorter than the red segment, and thus in either case the blue path is not shorter than the red segment. [Q.E.D.]

Proof of Lemma 5.

We assume the context of Lemma 6. In order to prove Lemma 5, it is assumed inductively that it holds for all (zero or more) rectangles placed before R by *PROV8*, and the induction step is performed by considering R.

All rectangles placed by *PROV8* before R is found are at least as thick as R. Therefore, by induction hypothesis, every marked vertex is marked with a length greater than $2 \times s$, so we have $l_{vert} > 2 \times m \times s$. Combining this with Lemma 6, we get

$l_R < 3 \times l_{blue} + m \times s < 3 \times l_{blue} + l_{vert}/2$. Thus we have $l_{rem} = 8 \times l_{blue} + l_{vert} - l_R > 8 \times l_{blue} + l_{vert} - (3 \times l_{blue} + l_{vert}/2) = 5 \times l_{blue} + l_{vert}/2 \geq 5 \times l_{blue} + m \times s$. Thus it remains to show that $5 \times l_{blue} + m \times s \geq 2 \times i \times s$.

Let us color the perimeter of R with green and red color as in the proof of Lemma 6. Thus the number i, ie. the number of corners of R which are going to be marked by *PROV8* during its next step, is equal to the number of corners of R which are endpoints either of two red segments, or of one red and one green, or of two green segments.

The various cases are considered. Case 1 is when there are no red segments. In Case 1 i is equal to the number of corners of R which touch two green segments. But no green segment touches more than one corner of R, and therefore the number of green segments is not less than $2 \times i$. On the other hand, no two green segments touch the same marked vertex, so in Case 1 it holds that $m \geq 2 \times i$, and hence $m \times s \geq 2 \times i \times s$. Therefore the lemma holds in Case 1.

In Case 2 there are both green and red segments. Let A and B be the endpoints of the side of R on which there is a marked vertex. Thus the side AB touches a rectangle placed previously by *PROV8*. Let R' be that rectangle. Since R' is maximal, the side AB cannot include a side of R' (otherwise it would be possible to extend R' so that it overlaps with R). Thus either A or B lies on the perimeter of R', and we conclude that $i \leq 3$. On the other hand, since there is at least one red segment, there is a blue path in the open region bounded by R from one of its sides to the opposite one, so $l_{blue} \geq s$. Putting everything together we obtain $5 \times l_{blue} + m \times s \geq 5 \times s + s \geq 2 \times i \times s$, and thus the lemma holds also in Case 2.

In the remaining case, Case 3, there are no green segments. If there are no red segments either, then we have i=0 and the lemma trivially holds. Otherwise, as in Case 2, it holds that $l_{blue} \geq s$, and therefore the lemma holds for $i \leq 2$. It remains to show that if $3 \leq i \leq 4$, then $l_{blue} \geq 2 \times s$. To prove this, two subcases are distinguished. (cf. the proof of Lemma 6).

The first subcase is when there is no isothetic line passing through R which does not touch some blue segment within R. In this case it is easy to conclude, as in the proof of Lemma 6, that $l_{blue} \geq 2 \times s$.

The second and last subcase is analogous as the one considered in Proposition 1, and we assume the context of that proposition. Since we are considering the case when $3 \leq i \leq 4$, it follows that there is at least one horizontal red segment above the horizontal line l, and at least one below l. Thus, by the same arguments as in the proof of Proposition 1, we get $l_{blue} \geq 2 \times s$ also in this subcase. [Q.E.D.]

MODIFIED THICKEST-FIRST PARTITIONS FOR HISTOGRAMS

In this section it is shown how to modify slightly the idea of thickest-first partitions in order to derive an approximation algorithm which for any histogram P, it produces a rectangular partition of P of length $< 2.42 \times M(P)$. All the partitions which are described in this section can be computed efficiently in time $O(n \log n)$ if the input polygon is a histogram [6]. Also, Theorem 4 is stated, which partly explains why we have chosen exactly this approximation algorithm. Throughout the rest of the paper it is assumed w.l.o.g. that all histograms have horizontal basis, lying at the bottom.

To go on we need some definitions. Let q be any positive real number. A rectangle is called q-rectangle, if the ratio of the length of its horizontal side divided by the length of its vertical side is equal to q. A rectangle R is q-thicker than a rectangle R' if within R

a larger q-rectangle can be drawn than within R'. q-thickest-first rectangular partitions are defined as thickest-first partitions, with the only difference that thickest is replaced by q-thickest. The following theorem can be proved:

Theorem 4: For all reals $q > 0$, let $f(q)$, resp. $f'(q)$, be the least real number, such that for every non-degenerate, resp. possibly degenerate, histogram P (with the orientation assumed above), every q-thickest-first partition of P is of length $\leq f(q) \times M(P)$, resp. $\leq f'(q) \times M(P)$ (such a real always exists, and therefore our definition is correct). Then it holds that $f(q) = \max(1+q, \frac{q+2}{q})$, and $f'(q) = \max(2q, \frac{q+2}{q})$.

Corollary 1:.
The function $f(q)$, respectively $f'(q)$, is minimized if $q = \sqrt{2}$, respectively $q = \frac{\sqrt{17}+1}{4} \approx 1.28$. Thus the corresponding values of f and f' are $1+\sqrt{2}$, respectively $\frac{\sqrt{17}+1}{2} \approx 2.56$.

If degeneracies occur, a slightly modified $\sqrt{2}$-thickest-first partition can be computed also in time $O(n \log n)$, so that it achieves the same worst-case approximation factor as for non-degenerate histograms, ie. $1+\sqrt{2}$. The new partitioning procedure can be recursively defined as follows. Let R be any $\sqrt{2}$-thickest rectangle within P. Draw R unless all three conditions stated below hold.

(a) On the top of R there is only one (concave) vertex of P, and that vertex is of 360 degrees.

(b) The horizontal side of R has length at least $(1+\sqrt{2})$ times the length of its vertical side.

(c) For each other (concave) vertex of P lying on the perimeter of P, the distance from the vertex to the basis of the histogram is less than one third of the length of the vertical side of R.

If all these conditions hold, then, instead of drawing R, draw a vertical segment connecting the degenerate vertex mentioned in condition (a) with the basis of the histogram. Process in the same way subhistograms in the partition induced by the drawn segments, until P is completely partitioned into rectangles.

THE MODIFIED PARTITIONS FOR HISTOGRAMS ARE SHORTER THAN 2.42 TIMES THE LENGTH OF MINIMUM LENGTH PARTITIONS

The proof is similar to the proof concerning the length of thickest-first partitions for general polygons, presented in a previous section, so we assume that context. Because of space considerations, only the case when the histogram is non-degenerate is considered in this paper, and the following corollary of Theorem 4 is proved:

Conclusion 2: For any non-degenerate histogram P (with horizontal basis, as assumed above), it holds that every $\sqrt{2}$-thickest-first partition of P is of length $< (1+\sqrt{2}) \times M(P)$.

Proof : We employ a procedure analogous to *PROV8*, which was used to prove Lemma 5. Let us call the new procedure *PROH*. The only difference between *PROV8* and *PROH* is that instead of drawing maximal thickest rectangles, *PROH* draws maximal $\sqrt{2}$-thickest rectangles. But for this difference, *PROH* draws with black color and erases blue segments exactly as *PROV8*. When the polygon is a histogram, then it is not difficult to see that the procedure never marks any vertices, because, as it is easily seen,

(i) for any histogram P, and for any maximal rectangle R within P, all corners of R lie

on the perimeter of P, and
(ii) if $R \neq P$, then R partitions P into two or several histograms, so condition (i) holds also in the recursive application of the method to resulting subfigures.

Therefore, assuming the same notation as in the context of the description of *PROV8*, it is enough to prove that $l_R < (1+\sqrt{2}) \times l_{blue}$. The same method is used as in the proof of Lemma 6.

In the continuation let s' be the minimum among (i) the length of the vertical side of R, and (ii) the length of the horizontal side of R divided by $\sqrt{2}$. With other words, s' is the length of the vertical side of a largest $\sqrt{2}$-rectangle lying within R.

Let us color brown all parts of the perimeter of R which do not lie on the boundary of P (in doing this, we assume that all blue segments remain visible, so that brown and blue segments may overlap). Note that because P is a histogram with horizontal basis, there is at most one vertical brown segment to the left and at most one to the right. A crucial observation, a counterpart of Observation 1, is the following:

Observation 3 : No horizontal brown segment is longer than $\sqrt{2} \times s'$. Also none of the (at most two) vertical brown segments is longer than s'.

Proof : Analogous as the proof of Observation 1. [Q.E.D.]

In the continuation of the proof the possible cases are considered. Case 1 is when every vertical line passing through the open region bounded by R touches or intersects some blue segment lying in the closed region bounded by R. In this case l_{blue} is not less than the length of the horizontal side of R, and thus we have $l_{blue} \geq \sqrt{2} \times s'$. By Observation 3 the total length of the vertical brown segments is $\leq 2 \times s'$, and therefore it is $\leq \sqrt{2} \times l_{blue}$. On the other hand, since R is maximal and the histogram P non-degenerate, the total length of all horizontal brown segments is less than the length of the horizontal side of R and, thus, $< l_{blue}$. Combining the two results we obtain that the total length of all brown segments is $< (1+\sqrt{2}) \times l_{blue}$, so Conclusion 2 holds in this case.

The remaining case, Case 2, is when there is some vertical line, say l', such that l' passes through the open region bounded by R, but it does not touch neither cross any blue segment in the closed region bounded by R.

To treat Case 2, let us change to red all parts of brown segments which do not lie on any blue segment. Also, let l'_{blue} be the total length of all (parts of) blue segments lying in the open region bounded by R. To finish the proof of Conclusion 2 it remains only to prove that in Case 2 the total length of the red segments is $< (1+\sqrt{2}) \times l'_{blue}$. In particular it is enough to show (A) that the total length of the vertical red segments is $< l'_{blue}$, and (B) that the total length of the horizontal red segments is $\leq l'_{blue}$. (A) follows from Proposition 1, by just turning 90 degrees the whole figure considered in the proof of Lemma 6. (B) is a counterpart of Proposition 2. For each horizontal red segment a rectangle R' is defined, such that the red segment is the top of R' and the basis of R' is on the basis of the histogram. By Observation 3 the length of the red segment divided by the length of the vertical side of R' is $\leq \sqrt{2}$. In this way the proof of (B) can be derived from the proof of Proposition 2. The details are easy and are therefore omitted. [Q.E.D.]

ACKNOWLEDGMENTS

I am indebted to Andrzej Lingas for many helpful discussions. I would also like to thank

Rolf Karlsson for some constructive comments.

REFERENCES

[1] Chazelle, B. and D. Dobkin, "Decomposing a Polygon into its Convex Parts", Proc. 11th ACM Symp. on Theory of Comput., 1979.

[2] Chew, L.P., and Drysdale, R.L., "Voronoi Diagrams Based on Convex Distance Functions", Proc. First ACM Symposium on Computational Geometry, Baltimore, June 1985.

[3] Fortune, S.J., "A Fast Algorithm for Polygon Containment by Translation", Proc. 12th ICALP, Nafplion, Greece, 1985.

[4] Gonzalez, T., and S.Q. Zheng, "Bounds for Partitioning Rectilinear Polygons" Proc. First ACM Symposium on Computational Geometry, Baltimore, June 1985.

[5] Hwang, F.K., "An $O(n \log n)$ Algorithm for Rectilinear Minimum Spanning Trees", JACM 26 (1979), 177-182.

[6] Levcopoulos, C., "Heuristics for Decomposing Polygons into Rectangles" (preliminary title), Ph.D. thesis, to appear, Linköping.

[7] Levcopoulos, C., and A. Lingas, "Bounds on the Length of Convex Partitions of Polygons", Proc. 4th Conf. on Found. of Software Technology and Theoretical Computer Science, Bangalore, India, 1984 (Springer Verlag).

[8] Lingas, A., "Heuristics for Minimum Edge Length Rectangular Partitions of Rectilinear Figures", Proc. 6th GI-Conference, Dortmund, January 1983 (Springer Verlag).

[9] Lloyd, E.L., "On Triangulations of a Set of Points in the Plane", Proc. of 18th IEEE Conf. on Foundations of Comp. Sc., Providence, 1977.

[10] Lingas, A., R.Y. Pinter, R.L. Rivest, and A. Shamir, "Minimum Edge Length Partitioning of Rectilinear Polygons", Proc. 20th Annual Allerton Conference, 1982.

[11] Lubiw, A., "Decomposing Polygonal Regions into Convex Quadrilaterals" Proc. First ACM Symp. on Comp. Geometry, Baltimore, 1985.

[12] Rivest, R., "The PI (Placement and Interconnect) System", Proc. 19th Design Automation Conference, June 1982.

AN ALGORITHM FOR THREE LAYER CHANNEL ROUTING USING RESTRICTED OVERLAP

Martin L. Brady and Donna J. Brown
Department of Electrical and Computer Engineering
University of Colorado, Boulder, CO. 80309

ABSTRACT

We show how certain types of overlap can be used to achieve improved solutions for the channel routing problem, with three layers of interconnect. Our algorithm uses at most $d/2 + O(\sqrt{d})$ tracks to route a two terminal net CRP, and all wires are within $O(\sqrt{d})$ of their optimal length.

1. INTRODUCTION

We describe briefly here a new model for the channel routing problem (CRP), allowing three layers for interconnection. In most previous channel routing algorithms, wires on different layers are allowed to intersect at a point, but can not run on top of one another (*overlap*), in order to avoid capacitive coupling. We relax this model and allow certain types of overlap for which capacitive coupling is limited.

Unit vertical overlap for *two* layers of interconnect has been studied by Gao and Hambrusch [GH85], and later by Berger and Leighton [L85], who achieve a channel width of $d + O(\sqrt{d})$. Our *three* layer algorithm is an extension of Berger and Leighton's strategy. We solve the two terminal net channel routing problem using $d/2 + O(\sqrt{d})$ tracks, and the wires have length within $O(\sqrt{d})$ of optimal. In comparison, the best nonoverlap three layer routing requires d tracks [PL84], while the best known algorithm for routing in three layers using arbitrary overlap requires only about $2/3d$ tracks [BraBr85a].

2. ROUTING STRATEGY

We outline here the general strategy for solving a two terminal net CRP of density d. The algorithm proceeds in two phases. In Phase I, $d/2 + O(\sqrt{d})$ tracks in the middle of the channel are used to form an approximate solution to the CRP. Each net is guaranteed to be routed to within $3k\sqrt{d}$ columns of its entry or exit terminal (k is a small constant). Phase II is divided into upper and lower subproblems. $O(\sqrt{d})$ tracks at the bottom (top) of the channel are used to connect each net to its lower (upper) terminal, completing the solution of the CRP. Figure 1 shows the regions of the channel in which each phase is active.

Phase I: The channel is partitioned into $\left\lceil C/(3k\sqrt{d}) \right\rceil$ blocks, each block containing $3k\sqrt{d}$

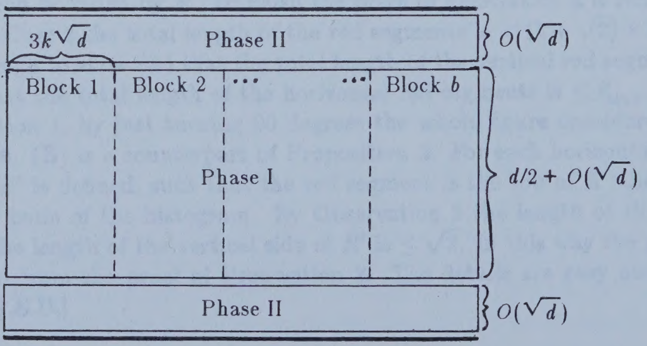


Figure 1. General Routing Strategy.

columns (see Figure 1). Each net is routed from the block in which it begins to the block in which it ends. The net need not be routed into the correct entry and exit columns; any column within the correct block is acceptable. (Thus, a net which begins and ends in the same block could be routed as a trivial net in Phase I.) Phase I routing is organized as follows. There are $\sqrt{d}/2k$ groups of $k\sqrt{d}$ consecutive tracks. Each of these tracks contains two distinct nets, one in layer 1, the other in layer 3, being routed horizontally. In addition, each of the $\sqrt{d}/2k$ groups are separated by two *free* tracks. Within each block of columns, the free tracks are used in the process of making vertical connections. Thus, a total of $d/2 + O(\sqrt{d})$ tracks are used in Phase I.

Phase II: All nets have been routed into the correct $3k\sqrt{d}$ column block. Thus the remaining upper and lower Phase II subproblems each have density at most $3k\sqrt{d}$. Here, we use the optimal *nonoverlap* routing algorithm of [PL84] to route the upper and lower Phase II subproblems using at most $3k\sqrt{d}$ tracks for each.

> *Theorem 1:* Any two terminal net CRP with density d can be routed in the three layer unit vertical overlap model using at most $d/2 + O(\sqrt{d})$ tracks, such that all wires have length within $O(\sqrt{d})$ of optimal.

The handling of multiterminal nets seems to be more difficult. The obvious extension of the above strategy results in a doubling of the number of tracks required in Phase I, leading to $d + O(\sqrt{d})$ tracks. But [BraBr85b] present a multiterminal net routing algorithm for three layers which requires only d tracks and uses no adjacent overlap at all. The ideas used in this paper extend also to a $3/7d + O(\sqrt{d})$ algorithm for four layers; this improvement, however, necessitates adjacent overlap of arbitrary length. (In [BraBr85b] four layer multiterminal net routing is is achieved using at most $\lceil d/2 \rceil + 4$ tracks using only nonadjacent overlap.)

REFERENCES

[BraBr85a] Brady, M. and D. J. Brown, "A 2/3d Algorithm for Three Layer Channel Routing With Overlap," draft, 1985.

[BraBr85b] Brady, M. and D. J. Brown, "Optimal Multilayer Channel Routing With Overlap," , draft, 1985.

[GH85] Gao, S. and S. Hambrusch, "Two-Layer Channel Routing with Vertical Unit-Length Overlap," draft, March 1985.

[L85] Leighton, F. T., personal communication

[PL84] Preparata, F. P. and W. Lipski, Jr., "Optimal Three-Layer Channel Routing," *IEEE Trans. on Computers* **C-33** pp. 427-437 (1984).

A Fast Algorithm For Single Row Routing*

Sangyong Han^{+} and Sartaj Sahni

University of Minnesota

Abstract: We develop a fast algorithm for optimal single row routing. Experimental results show that our algorithm is significantly faster than previously known algorithms.

Keywords and Phrases: Single Row Routing, Complexity

1. Introduction

So [So74] has proposed a systematic approach to the interconnection problem of large multilayer printed circuit boards in which pins and feedthroughs are uniformly spaced on a rectangular grid. This approach consists of a systematic decomposition of the general multilayer wiring problem into a number of independent single layer, single row routing problems. There are five phases in this decomposition ([TING78] and [STEV79]):

1. Via assignment
2. Linear placement of via columns
3. Layering
4. Single row routing
5. Via elimination

In this paper, we are concerned only with the fourth phase : Single row routing. In the single row routing problem, we are given a set V=$\{1, 2, .., n\}$ of n nodes that are evenly spaced along a straight line ; and a set L = $\{N_1, N_2, ..., N_m\}$ of nets. Each net represents a set of nodes that are to be made electrically equivalent. The nets satisfy the following conditions:

(i) $N_i \cap N_j = \emptyset \quad , i \neq j$

(ii) $\bigcup_{i=1}^{m} N_i = \{1, 2, ..., n\}$

$j \epsilon N_i$ is a *touch point* of net i. The nets are to be realized in a single layer by the use of non-overlapping wires that are composed solely of horizontal and vertical segments. Figure 1.1(a) shows some of the legal ways to realize the net $\{1, 3, 6, 9\}$. The wire layout must satisfy the additional requirement that a vertical cut made at any point along the axis formed by the nodes can intersect at most one horizontal segment from each net. Thus the wiring of Figure 1.1(d) is illegal.

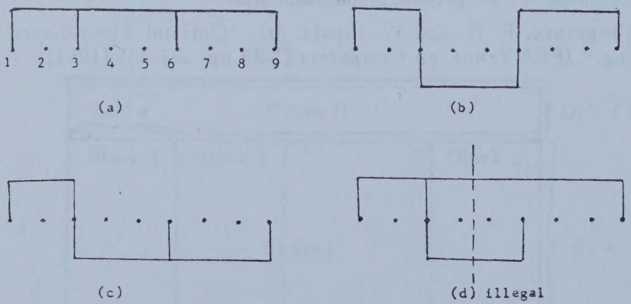


Figure 1.1

---

*This Research was supported in part by the National Science Foundation under grant MCS-8305567

+ Dr Han is presently with IBM, Kingston

The area above the line of nodes is called the *upper street* while that below this line is the *lower street.* Each street has *tracks* that run parallel to the line of nodes (Figure 1.2). Horizontal wire segments must be layed in tracks and no track may hold more than one wire segment at any point (of course, several non overlapping wire segments may be packed into the same track). Let K_u and K_l , respectively, denote the number of tracks in the upper and lower streets. Figure 1.2 shows one way to realize the nets $(N_1, N_2, \ldots, N_5) = (\{1,7\}, \{2,8\}, \{3,6\}, \{4,9\}, \{5,10\})$ when $K_u = 2$ and $K_l = 3$.

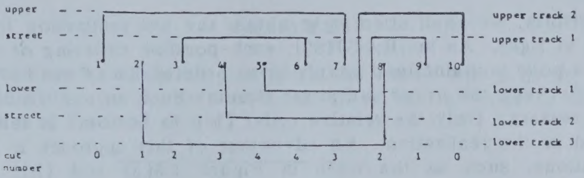


Figure 1.2

In this paper, we are specifically concerned with obtaining net realizations (when they exist) given V (node set), L (net set), K_u (upper street capacity), and K_l (lower street capacity).

This problem has been studied before. Kuh et al. [KUH79] and Tsukiyama et al. [TSUK80] developed necessary and sufficient conditions for a net set to be realizable. In [KUH79], a simple construction is used to show that when K_u and K_l are sufficiently large, L is realizable. This construction, however, results in an algorithm of complexity O(m!n) where $|V| = n$ and $|L| = m$. Raghavan and Sahni [RAGH82] have developed an algorithm of complexity O(k!*k*n*log k) where $k = K_u + K_l$ for this problem. This algorithm is quite practical when k is small but impractical when k is large. Han and Sahni[HAN83], have developed fast linear time algorithms for the cases $K_u \le 3$, $K_l \le 3$ and $K_u = 1$, $K_l \ge 1$. These algorithms are substantially faster than that of [RAGH82].

In this paper, we extend our work of [HAN83] to the case of arbitrary K_u and K_l. While we explicitly consider only the case $K_u = K_l$, our algoithm is easily generalized to the case of $K_u \neq K_l$. Experimentation with our algorithm indicates that it is eight to more than a hundred times faster than the algorithm of [RAGH82]. Also, it requires considerably less computer memory.

Before presenting the algorithms, we introduce some notation. Following [RAGH82], nodes are classified by type :

(a) Node i is of *type B* if it is the left extreme node of a net. I.e., it is the *beginning* node for some net. The type B nodes in Figure 1.2 are 1, 2, 3, 4, and 5.

(b) A *type E* node is a node at which a net i ends. I.e., it is a right extreme node for some net. The type E nodes in Figure 1.2 are 6, 7, 8, 9, and 10.

(c) A *middle* or "*non extreme*" node of a net i is a *type M* node. If a and b are, respectively, the B and E nodes of a net i, then $N_i - \{a, b\}$ is the set of middle or type M nodes for this net.

The *cut number* of a node i is the number of nets covering that node. I.e., the number of nets between whose extreme nodes i lies. The net (if any) with touch point i is not included in this count. Cut numbers for the nodes of Figure 1.2 are given in this figure.

A *j-zone* ([TSUK80]) is a segment of the node axis beginning at a B type node with cut number j and ending at the next E type node with this cut number. Thus, all cut numbers within a *j-zone* are greater than or equal to k. The example of Figure 1.2 has the following zones :

0 - zone : nodes 1 through 10
1 - zone : nodes 2 through 9
2 - zone : nodes 3 through 8
3 - zone : nodes 4 through 7
4 - zone : nodes 5 through 6

In general, of course, a net list may have several *j*-zones for any given *j*.

In our algorithm, we shall attempt to obtain the net realization by processing the nodes from left to right. As in [RAGH82], each possible ordering of nets that can be encountered by a node is maintained simply as an ordered list of net indices. There is no explicit division between the upper and lower streets. Such an explicit division is neither desirable nor necessary. Only the relative order (top to bottom) is relevant; this same order is reflected in the realization. An advantage of this approach is that functionally equivalent situations, such as the ones in Figure 1.3(a) and (b), are not treated separately.

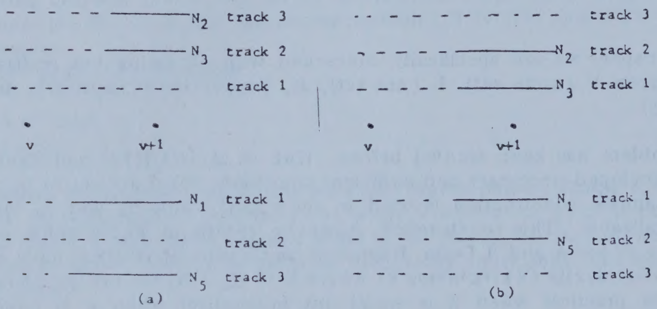


Figure 1.3

2.Routing with k tracks per street.

As remarked above, our algorithm will build feasible net pemutations at each node. This is done considering the nodes left to right. So, suppose σ is a feasible pemutation coming into node i (i.e., coming from the left of i). Consider the two cases: (a) cut number (i) $\leq$ k and (b) cut number (i) $>$ k.

When the cut number at i is $\leq$ k and i is of type M or E, then the net in σ with touch point i can connect to i regardless of the position of this net in σ (see Figure 2.1, for the case k $=$ 4).

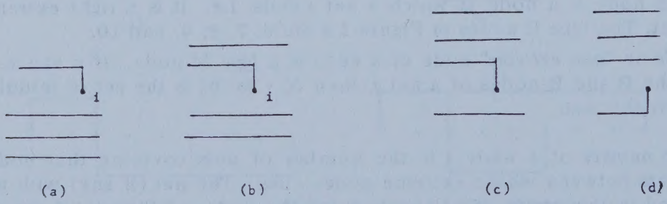


Figure 2.1

If i is of type B, then the net beginning at i can enter the permutation σ at any point (see Figure 2.2). Hence, if there are no (k+ 1)-zones, then the net set can be realized by making arbitrary decisions at nodes of type B. In the remainder of our discussion, we therefore assume that there exist some (k+ 1)-zones (and possibly some (k+ 2), ..., (2k-1)-zones too).

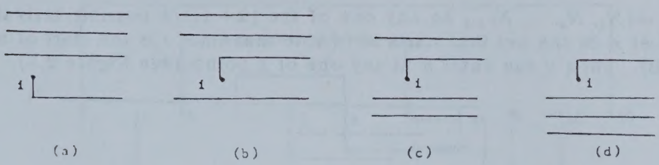


Figure 2.2

When the cut number at i is $k+j$, $1 \leq j < k$ and i is of type E or M, then the net in σ with touch point i can connect to i iff it is the q^{th} net in σ for some q, $j < q \leq k+1$ (Figure 2.3).

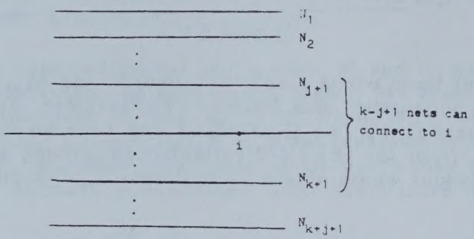


Figure 2.3

If i is of type B, then the net beginning at i may enter σ at any of the positions q, $j < q \leq k+1$ (Figure 2.4).

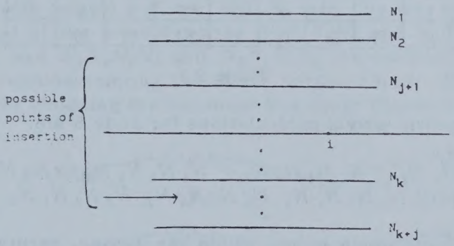


Figure 2.4

Let i be the start of the first (k+1)-zone. Exactly k+1 nets arrive at node i from the left. Since each node to the left of node i has a cut number $\leq$k, all (k+1)! permutations of these k+1 nets are feasible arrival permutations. However, when $K_u = K_l$ (which is the case we are explicitly considering here), these (k+1)! permutations may be paired into (k+1)!/2 groups such that the pair in each group is symmetric. We need therefore consider only (k+1)!/2 non symmetric arrival permutations at i.

Example 2.1: Let k=2 and assume that $\{N_1, N_2, N_3\}$ is the set of nets arriving at node i. Node i is the start of the first 3-zone. All 6 permutations, $N_1N_2N_3$, $N_1N_3N_2$, $N_3N_1N_2$, $N_2N_1N_3$, $N_2N_3N_1$, and $N_3N_2N_1$ are feasible arrival permutations. However, $N_1N_2N_3$ and $N_3N_2N_1$ are symmetric. So also are $N_2N_3N_1$ and $N_1N_3N_2$; and $N_3N_1N_2$ and $N_2N_1N_3$. So, we need consider only the 3 non symmetric arrival permutations $N_1N_2N_3$, $N_1N_3N_2$, $N_2N_1N_3$.□

Let $\sigma = (N_1, N_2, \ldots, N_{k+1})$ be any one of the (k+ 1)!/2 possible arrival permutations at node i. Let x be the net that starts at i (note that since i is the start of a zone, it must be of type B). Then x can enter σ at any one of k points (see Figure 2.5).

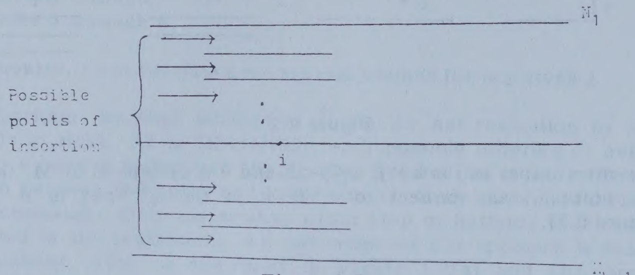


Figure 2.5

Regardless of which of these points x enters at, nets N_1 and N_{k+1} remain the outermost nets in the resulting permutation that leaves i. Furthermore, N_1 and N_{k+1} remain the outermost nets throughout this (k+ 1)-zone and may have no touch points within the (k+ 1)-zone. Hence, from the (k+ 1)!/2 permutations arriving at σ we may delete as infeasible, all permutations whose outermost nets have a touch point within the (k+ 1)-zone.

Example 2.2 : Let k be 3 and let the net list be as given in Table 2.1. This net list consists of 11 nodes and 5 nets. There is only one 4-zone (from node 5 to node 7) and no 5-zone. Two nets (N_3 and N_4) have touch points inside this 4-zone.

NODE	1	2	3	4	5	6	7	8	9	10	11
NET	1	2	3	4	5	4	3	4	5	2	1
TYPE	B	B	B	B	B	M	E	E	E	E	E

Table 2.1

The 12 possible asymmetric arrival permutations for node 5 are:

$N_1 N_2 N_3 N_4$, $N_1 N_2 N_4 N_3$, $N_1 N_4 N_2 N_3$, $N_4 N_1 N_2 N_3$, $N_1 N_3 N_2 N_4$, $N_1 N_3 N_4 N_2$, $N_1 N_4 N_3 N_2$, $N_4 N_1 N_3 N_2$, $N_3 N_1 N_2 N_4$, $N_3 N_1 N_4 N_2$, $N_3 N_4 N_1 N_2$, $N_4 N_3 N_1 N_2$.

Since N_3 and N_4 have touch points within the 4-zone, permutations with these nets at position 1 or 4 are infeasible. Eliminating these infeasible arrival permutations leaves us with just two candidates ($N_1 N_3 N_4 N_2$ and $N_1 N_4 N_3 N_2$) for the arrival permutation at node 5. Both of these yield the same set of emerging permutations ($\{N_1 N_5 N_4 N_2, N_1 N_4 N_5 N_2\}$) from the 4-zone. Since, there are no other 4-zones, we may arbitrarily pick one of the remaining feasible candidates (*say* $N_1 N_3 N_4 N_2$) and obtain the routing. In order to get this permutation at node 5, we need to place N_2 below N_1 at 2, N_3 between N_1 and N_2 at 3 and N_4 between N_3 and N_2 at 4. Regardless of whether N_5 is placed above or below N_4 at 5, the routing can be completed. Suppose we elect to place N_5 between N_4 and N_2. The resulting layout is shown in Figure 2.6.□

The above elimination process is easily extended to (k+ j)-zones for $j>1$. A further reduction in the number of arrival permutations one need consider at node i results from the following observation. Suppose that σ contains p nets that have no touch point in the (k+ 1)-zone and terminate before the start of the next (k+ 1)-zone. Then the relative ordering of these p nets is immaterial. In fact, the only arrival permutations that need be considered may be obtained from the (k+ 1-p)!/2 asymmetric permutations of the remaining k+ 1-p nets in the following way. Start with any permutation σ of the remaining (k+ 1-p) nets. Create p+ 1 new permutations by adding any j of the p nets to the top and the remaining p-j to the bottom of σ, $0 \leq j \leq$ p.

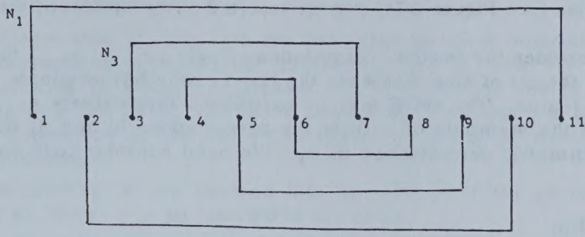


Figure 2.6

Example 2.3 : In this example, we have a net list which has 8 nets and 16 nodes and $K_u = K_l = 3$. Net list L is as follows; $N_1 = \{1,7\}$, $N_2 = \{2,12\}$, $N_3 = \{3,6\}$, $N_4 = \{4,8\}$, $N_5 = \{5,13\}$, $N_6 = \{9,14\}$, $N_7 = \{10,15\}$, $N_8 = \{11,16\}$.

Among the four nets entering the first 4-zone, N_1 and N_4 have no touch points in this zone and terminate before the next 4-zone begins. We begin with the permutation $N_3 N_2$ and add the two nets, N_1 and N_4, to it. From N_3N_2 we generate $N_1N_4N_3N_2$, $N_4N_3N_2N_1$ and $N_3N_2N_1N_4$. Actually $N_3N_2N_1N_4$ is not generated at all because it is infeasible (see Example 2.2). Therefore at the start of the first 4-zone we have only two permutations instead of 4!/2=12 permutations. Here the relative order of N_1 and N_4 is immaterial.

At the end of the first 4-zone N_3 is gone and N_5 is added. From $N_1N_4N_3N_2$ we obtain $N_1N_5N_4N_2$ and $N_1N_4N_5N_2$. From $N_4N_3N_2N_1$, $N_4N_5N_2N_1$ and $N_4N_2N_5N_1$ are obtained. At the start of the second 4-zone N_1 and N_4 are gone and N_6 and N_7 are new nets both of which have no touch points in this zone and terminate before the next 4-zone begins. If we delete N_1 and N_4 from the permutations coming from the previous 4-zone, only one permutation, N_5N_2, will remain. If we add N_6 and N_7 we will have $N_6N_5N_2N_7$ and $N_5N_2N_7N_6$. At the end of the second 4-zone $N_6N_8N_5N_7$ and $N_6N_5N_8N_7$ are derived from the first permutation and $N_5N_8N_7N_6$ and $N_5N_7N_8N_6$ are obtained from the second one. Figure 2.7 shows all relations among the above permutations. The layout shown in Figure 2.8 was obtained by choosing the top most link from Figure 2.7. □

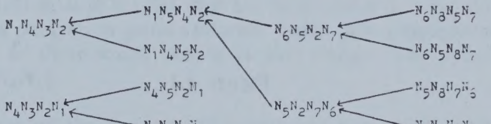


Figure 2.7

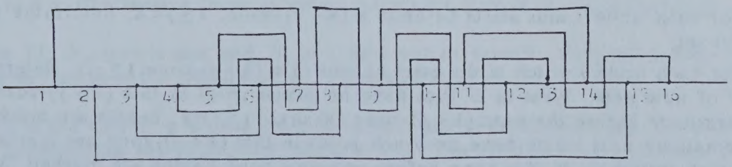


Figure 2.8

The above permutation reduction procedure may also be applied to $(k+j)$-zones, for $j>1$. Another observation useful in reducing the number of permutations to be considered is: A net which begins a $(k+i)$-zone, $i \geq 1$, terminates in that zone before any other zone begins (see Figure 2.7), may be inserted at the middle of the permutation.

Finally, consider the feasible permutations $S=\{\sigma_1, \sigma_2, ..., \sigma_{k+1}\}$ that leave a (k+1)-zone. Let x be the set of nets that leave the (k+1)-zone but terminate before the start of the next (k+1)-zone. The set S may be partitioned into subsets $A_1, A_2, \cdots$ such that the deletion of the members of x from the permutations in any A_j leaves a multiset of identical or symmetric permutations in A_j. We need consider only one σ from each of the A_is.

3. The Algorithm

Definitions: Let node i be the start or end of some (k+j)-zone, $1 \leq j < k$. The set S of nets that arrive at i from the left but do not terminate at i is called the *entering net set.* Let q, $q<i$, be the largest q that is the start or end of a (k+r)-zone, $1 \leq r < k$. If no such q exists, set q to zero. Let T be the entering set for q. $T=\phi$ if $q=0$. Let v be the net (if any) that terminates at i. The nets in $S\text{-}T\text{-}\{v\}$ are considered *new nets* at i.

Example 3.1 : Consider the single row routing instance with k = 3 and net list; $N_1 = \{1,7\}$, $N_2 = \{2,6\}$, $N_3 = \{3,11\}$, $N_4 = \{4,13\}$, $N_5 = \{5,12\}$, $N_6 = \{8,14\}$, $N_7 = \{9,15\}$, $N_8 = \{10,16\}$. The nets are shown pictorially in Figure 3.1. This figure also shows the start and end of all the 4 and 5 zones. $\{N_1, N_2, N_3, N_4\}$, $\{N_1, N_3, N_4, N_5\}$, $\{N_3, N_4, N_5, N_6\}$, $\{N_3, N_4, N_5, N_6, N_7\}$, $\{N_4, N_5, N_6, N_7, N_8\}$ and $\{N_4, N_6, N_7, N_8\}$ are, respectively, the entering net sets at nodes 5, 6, 9, 10, 11 and 12. The set of new nets at each of these nodes is, respectively, $\{N_1, N_2, N_3, N_4\}$, $\{N_5\}$, $\{N_6\}$, $\{N_7\}$ and $\{N_8\}$. □

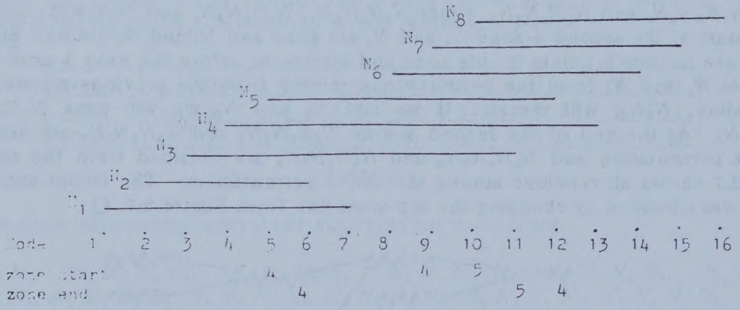


Figure 3.1

Our single row routing algorithm consists of the following steps.

1. For each node i that starts or ends a (k+j)-zone, $1 \leq j < k$, determine the entering net set.
2. For each node i which is the start or end of a (k+j)-zone, $1 \leq j$,k, determine the set U of new nets. Nets in U that have no touch point in this (k+j)-zone and which terminate before the next (k+l)-zone (if any), $1 \leq l < k$, begins are marked "2". The remaining nets which have no touch point in this (k+j)-zone are marked "0". Nets which terminate in this zone before any new zone begins are marked "3". All other nets are marked "1".
3. Generate permutations at each node i which begins or ends a (k+j)-zone, $1 \leq j < k$ as below. The nodes are considered left to right.
   (a) Let Π be the set of all permutations remaining at the most recent start or end of a (k+q)-zone, $1 \leq q < k$. $\Pi = \phi$ if we are at the start of the first (k+1)-zone. From the permutations in Π, remove all nets that terminate before or at node i. Eliminate duplicate and symmetric permutations in Π. Into the permutations of Π insert all new nets of type "0" or "1" getting all feasible (i.e., without

violating the routing constraints) new permutations.

(b) Insert all new type "2" nets into the permutations of a) without violating the routing constraints. Let p be the number of type "2" nets. We add any q of these beginning at position j (position $j+1$ if this is the end of a zone). The remaining $p-q$ are added just above the $(j-1)^{th}$ (position j if the end of the zone) net from the bottom, $0 \leq q \leq p$. Thus each initial permutation results in p+1 permutations.

(c) New nets marked "3" are inserted into the middle of the permutations generated in b). These may be inserted in any order.

(d) If this is the start of a zone, then eliminate all permutations that have a net which has a touch point within this zone and which is either in position j from the top or j from the bottom. In our program step d is coupled with steps (a), (b) and (c) so that permutations to be eliminated are not generated at all.

4. If the permutation set becomes null at the start or end of any (k+ j)-zone, $1 \leq j < k$ above, then no layout using k tracks exists. Otherwise, a layout exists.

An example will illustrate the details of the above procedure.

Example 3.2 : Consider the instance of Example 3.1. Node 5 is the start of the first 4-zone. So the set Π of remaining permutations is ϕ. The new nets N_1, N_2, N_3, and N_4 are, respectively, of type 2, 3, 0, and 0. Inserting nets N_3 and N_4 into the permutations of Π yields the permutations N_3N_4 and N_4N_3. Deleting symmetric permutations leaves us with the lone permutation N_3N_4. IN step 3(b), N_1 is inserted and the permutation $N_1N_3N_4$ and $N_3N_4N_1$ are obtained. Finally, in step 3(c) the type 3 net N_2 is inserted to yield the permutations $N_1N_2N_3N_4$ (or $N_1N_3N_2N_4$) and $N_3N_2N_4N_1$ (or $N_3N_4N_2N_1$).

At node 6, the set Π is $\{N_1N_2N_3N_4, N_3N_2N_4N_1\}$. N_2 terminates at this node. Deleting N_2 yields $\Pi = \{N_1N_3N_4, N_3N_4N_1\}$. The new net N_5 is of type 1. Inserting it at all possible feasible points yields the permutation set $\{N_1N_5N_3N_4, N_1N_3N_5N_4, N_3N_5N_4N_1, N_3N_4N_5N_1\}$.

At node 9, we delete N_1 from the remaining permutations at node 6. After removing duplicates, we get $\{N_5N_3N_4, N_3N_5N_4, N_3N_4N_5\}$. The new net N_6 is of type 2 and the resulting permutation set is $\{N_6N_5N_3N_4, N_5N_3N_4N_6, N_6N_3N_5N_4, N_3N_5N_4N_6, N_6N_3N_4N_5, N_3N_4N_5N_6\}$. However, nets N_3 and N_5 have a touch point in this 4-zone and are in position 1 from the top or bottom in the second, fourth, fifth and sixth permutations. These are eliminated as from these N_3 and N_5 cannot connect to their touch points in the 4-zone. The remaining permutation set is $\{N_6N_5N_3N_4, N_6N_3N_5N_4\}$.

Node 10 is the start of the 5-zone. The new net here is N_7 and this is of type 2. Inserting it into the permutation set gives us $\{N_6N_7N_5N_3N_4, N_6N_7N_3N_5N_4, N_6N_3N_5N_7N_4\}$. Net 3 has a touch point in this 5-zone and is in position 2 from the top or bottom in permutations 1 and 4. Eliminating these gives us the set $\{N_6N_5N_3N_7N_4, N_6N_7N_3N_5N_4\}$.

At node 11, N_3 terminates and N_8 is a new net of type 2. Following node 11 the remaining permutation set is $\{N_6N_5N_8N_7N_4, N_6N_7N_8N_5N_4\}$.

Node 12 is the end of a 4-zone. Net N_5 terminates here. Eliminating this net leaves us with the permutation set $\{N_6N_8N_7N_4, N_6N_7N_8N_4\}$. The relation between these permutations is shown in Figure 3.2 and a realization using the top most links is given in Figure 3.3. □

4. Experimental Results

The performance of the algorithm described in section 3 was compared to that of the algorithm given in [RAGH82]. For the algorithm of [RAGH82], we used the Pascal code provided to us by Dr. Raghavan. We programmed our algorithm in Pascal. Both algorithms were run on a VAX 11/780 and the observed running times are shown in Table

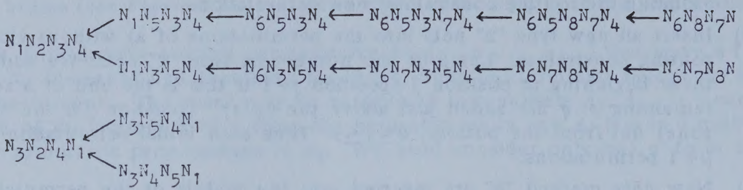

$N_1N_5N_3N_4$
$N_6N_5N_3N_4$
$N_6N_5N_3N_7N_4$
$N_6N_5N_8N_7N_4$
$N_6N_8N_7N$
$N_1N_2N_3N_4$
$N_1N_3N_5N_4$
$N_6N_3N_5N_4$
$N_6N_7N_3N_5N_4$
$N_6N_7N_8N_5N_4$
$N_6N_7N_8N$
$N_3N_5N_4N_1$
$N_3N_2N_4N_1$
$N_3N_4N_5N_1$

Figure 3.2

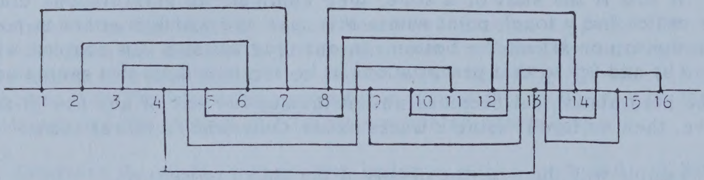

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

Figure 3.3

4.1.

Random single row routing instances with either 99 or 60 nodes and 20 nets were generated. The computing times reported are for layout runs with optimal K_s and K_l values. As can be seen from the table, the algorithm of [RAGH82] could not complete all the instances because of time and space limitations. On instances which it did complete the layout, our algorithm took one eighth to less than one hundredth of the time.

We also tested a single row routing instance that appears in the [KUH79]. On this instance our algorithm took 1300 ms while that of [RAGH82] took 103141 ms. For this run, $K_s = 5$ and $K_l = 4$.

Number of nodes	Maximum cut number	K_s and K_l	Algorithm in [RAGH82]	Our Algorithm
99	7	5,5	111283	466
99	7	5,5	95000	400
60	8	6,6	599100	350
60	8	5,5	70583	1816
60	8	5,5	64467	366
60	8	5,4	37000	15733
60	9	6,6	**	983
60	9	7,7	**	19333
60	9	7,7	**	10700
60	9	6,6	**	1500
60	9	6,6	**	11233

times are in milliseconds

**...could not complete because of excessive time and space requirements.

Table 4.1

5.CONCLUSIONS

We have developed a relatively simple algorithm for single row routing. Experimental results indicate it is significantly faster than all previously known algorithms for this problem.

REFERENCES

[BREU72] M.A. Breuer [ed.], Design Automation of Digital Systems. Prentice-Hall 1972.

[HAN83] S.Han and S.Sahni, "Single Row Routing in Narrow Streets", *Technical Report 83-8, University of Minnesota,* to appear at *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems.*

[KUH79] E.S. Kuh, T.Kashiwabara and T.Fujisawa, "On Optimum Single Row Routing", *IEEE Transactions on Circuits and Systems,* Vol. CAS 26, No.6, June 1979, pp. 361-368.

[RAGH82] R.Raghavan and S.Sahni, "Optimal single row router", *19th ACM IEEE Design Automation Conference,* pp. 38-45.

[STEV79] K.Stevens and W.Van Cleemput, "Global via elimination in a generalized routing environment", *IEEE ISCAS Proceedings,* 1979, pp.689-692.

[SO74] H.C. So, "Some theoretical results on the routing of multilayer printed-wiring boards", *IEEE Symposium on Circuits and Systems* 1974, pp. 296-303.

[TING76] B.S.Ting, E.S. Kuh and I.Shirakawa, "The multilayer routing problem:Algorithms and Necessary and Sufficient Conditions for the single row single layer case", *IEEE Transactions on Circuits and Systems,* Vol. CAS 23, No.12, December 1976, pp. 768-778.

[TING78] B.S.Ting, E.S. Kuh, "An approach to the routing of multilayer printed circuit boards", *IEEE Symposium on circuits and systems* 1978, pp. 902-911

[TSUK80] S. Tsukiyama, E.S.Kuh and I.Shirakawa, "An algorithm for single row routing with prescribed street congestions", *IEEE Transactions on circuits and systems,* VOL. CAS 27, No.9, September 1980, pp.765-771

PLACEMENT THROUGH PAIRWISE INTERCHANGE AMONG CONNECTED AND UNCONNECTED LOGIC MODULES

SOUMITRA NANDY[1] and L.M.PATNAIK
Indian Institute of Science
Bangalore 560 012, INDIA

ABSTRACT- In this paper an algorithm for placement of N logic modules in a circuit is presented, based on trial pairwise interchange among connected unconnected pairs of logic modules. The objective function for the algorithm presented is the minimization of total wirelength (TWL). In this algorithm the trial number of interchanges are limited to only connected or unconnected pairs of logic modules at a time, which comprise only a subset of the total of ${}^{N}C_2$ pairs of logic modules. as a result, this algorithm performs better than other placement algorithms, based on pairwise interchange of modules both in terms of total wirelength in the final placement and computation time required to achieve this.

Design of integrated circuits comprises placing box-like rectangular blocks (called logic modules) of various dimensions in a plane and connecting them by wires according to a given set of interconnection patterns. Formally the module placement problem consists of finding the optimal placement of modules in the plane with respect to some norm (objective function) on the interconnections, such as minimal weighted wirelength.

In this paper, we present a placement algorithm based on pairwise interchange of logic modules. This algorithm achieves a better placement than the other placement techniques reported in the literature[1] [2] (those that employ trial interchange of all the ${}^{N}C_2$ module pairs in a circuit consisting of N logic modules) both in terms of total wirelength and computation time. In this algorithm, the set of Unconnected Pairs of Modules, UPM (which do not have a net connecting them) and the set of Connected Pairs of Modules, CPM (which are electrically common and have a net connecting them) are identified from the input netlist. Pairwise trial interchange is carried out on UPM until it fails to improve the placement any further in terms of wirelength or the maximum of N iterations of pairwise trial interchanges are over, where N is the maximum number of modules. The same is carried out with CPM also. The entire procedure is then repeated until one of the following terminating conditions is encountered:(a) the specified number of 100 iterations is over (b) the maximum computation time of 1800 seconds is elapsed (c) the placement cannot be improved any further.

---

[1] The author is presently a visiting scholar at the Department of Computer Science, State University of New York at Stony Brook.

By limiting the trial pairwise interchanges either to UPM or CPM, only a subset of the ${}^{N}C_2$ pairs is considered as candidate for an iteration. In such a scheme , the members of the UPM are separated by larger distances over every iteration on UPM and the members of the CPM are drawn closer with every iteration on CPM, resulting in a placement better than the placement reported in [1],[2],[3]. The algorithm is presented in a pseudo-pascal language in the following lines:

```
PROCEDURE Placement;
    BEGIN
        determine UPM and CPM;
        set initial random placement of modules;
        REPEAT
            REPEAT pairwise interchange of UPM
            UNTIL no further decrease in total wirelength or
                the maximum of N iterations is over;
            REPEAT pairwise interchange of CPM
            UNTIL no further decrease in total wirelength or
                the maximum of N iterations is over;
        UNTIL terminating conditions;
    END;
```

In conclusion, an algorithm to place logic modules in a circuit layout has been described in this paper. The main results are, (i) this algorithm gives a better performance with regard to total wirelength as criterion compared to other algorithms based on trial interchange of modules (ii) this algorithm is computationally less expensive than the other algorithms and has a time complexity of O(N^4).

REFERENCES

1. Cote, L.C., and Patel, A.M.,The Interchange Algorithms for Circuit Placement, Proc. Des. Aut. Conf. 1980, pp 528-534.

2. Nandy,S.K and Patnaik,L.M., A Hybrid Technique for Placement of Logic Modules, Proc. IEEE Intl. conf. on Comput.,Systems and Signal Processing, 1984, pp 1460-1463.

3. k.Ueda, T.Komatsubara and T.Hasaka, A Parallel Processing Approach for Logic Module Placement, IEEE Trans. on Computer-aided Design of ICs and Systems, Vol. CAD-2, No.1, pp. 39-47, Jan 1983.

OPTIMAL HOP-BY-HOP FLOW CONTROL IN COMPUTER NETWORKS

Zvi Rosberg *

Inder S. Gopal

IBM T.J. Watson Research Center

Yorktown Heights, N.Y 10598

Abstract

The problem of the hop-by-hop flow control in a computer network is formulated as a Markov decision process with a cost function composed of the delay of the messages and the buffer constraints. The optimal control is shown to have a linear truncated structure and the explicit form is found when the arrival process of the messages is a Bernoulli process. For a renewal arrival process, the long-run average cost of any policy with a linear truncated structure is expressed by a set of linear equations.

1. Introduction

Flow control mechanisms [GeKl], [Sc] are employed in computer networks in order to prevent the overloading of system resources and the consequent excessive time delays and buffer overflows. These mechanisms are classified in [GeKl] into four levels, the session level, the network access level, the end-to-end level, and the hop-by-hop level. The purposes of each level are complementary and in most typical networks more than one level exists. The focus of this paper, however, is the hop-by-hop level, the purpose of which is to maintain a smooth flow of traffic between two adjacent nodes, and to avoid local buffer congestion. Note that, at this level, we do not have the ability of restricting traffic input into the network.

The specific hop-by-hop scheme studied in this paper is the following. Consider two adjacent nodes, and for the purpose of the explanation, assume that one node is the transmitting node and the other the receiving node. The transmitting node maintains at all times a counter which indicates the number of messages that the receiver is prepared to accept. We shall use the term "window" to refer to this counter. Whenever it transmits a message, it decrements the counter by one. When the counter reaches zero, it is not permitted to transmit any more messages. The receiver, periodically sends to the transmitter a message indicating the new number of messages that it is prepared to accept (a new "window"), and the transmitter upon receipt of this message updates its counter. If the receiver has indicated to the transmitter that it is prepared to accept a certain number of messages, it is required to set aside an amount of storage in its receive buffers, sufficient to store that number of messages. Once the receiver has received the message and forwarded it to the next node on the message's path, it then releases the buffer storage used by that message.

In making the decision on the window size, the receiver has information on the history of message transmissions by the transmitter and on the number of messages waiting at the transmitter. The latter piece of information is sent by the transmitter to the receiver in the header of one of the messages that it transmits. However, because of transmission propagation or queueing delays on the link, the information is typically outdated when it reaches the receiver and does not reflect the current number of messages present at the transmitter.

---

* The author is on leave from the Technion-IIT, Department of computer Science, Haifa 32000, Israel.

The scheme described above is analogous to the hop-by-hop mechanisms used in real networks such as GMDNET and TYMNET [GeKl]. The mechanism may be applied once for all traffic that flows over a given link or may be applied individually for each virtual circuit that shares that link. We assume a single receive buffer pool that is completely shared by all links or virtual circuits that pass through the receiver.

There are clearly two conflicting forces that govern the receiver's actions. On the one hand, it is beneficial to commit to accept a large number of messages as this will reduce the possibility of the transmitter having to queue messages while awaiting permission to transmit from the receiver. On the other hand, if the receiver commits itself to receive too many messages it may find itself setting aside an excessive amount of storage, thereby preventing other links or virtual circuits from using that storage. The purpose of this paper is to understand this trade-off and to find the best strategy that the receiver should adopt. For a survey of previous and related studies, the reader is referred to [GeKl] or [Sc].

2. The Model

Consider two adjacent nodes and, as before, assume one is a transmitter (TR) and the other a receiver (RC). Assume further that all messages in the system consists of packets of fixed length. In order to preserve analytical tractability, we shall assume that both TR and RC operate in synchronous, tightly regulated fashions. In particular, we shall assume that time is divided into slots, a slot being exactly long enough to transmit a single packet. A packet transmission may begin only on a slot boundary. We shall also assume that the window allocation process proceeds in "phases", a phase being a fixed predetermined number of slots (say, T slots). Prior to the start of every phase RC informs TR of the window that it has chosen for that phase. This number represents the maximum number of packets that TR may transmit in that phase and also the number of buffer units of storage that RC must set aside for that phase. Clearly, the window size in any phase must be smaller than T.

In the first packet that TR transmits in a phase, it includes the current number of packets that it has awaiting transmission. This information is used by the receiver in making the window size decision at the start of the next phase. The receiver computes the window size and informs the transmitter of its decision through a control packet. This information must reach TR prior to the start of the next phase. Thus, T must be at least as the sum of the propagation time from TR to RC, the computational time required for RC to reach a decision, and the propagation time from RC to TR.

We assume that packet generation times forms a renewal process. That is, the number of packets generated during slot i, X_i, $i = 1,2,\ldots$ are i.i.d r.v.'s with a general distribution function F and a first moment λ. We assume that the arrival process is independent of the system state and flow control actions. We shall also place a further restriction on the model that the messages that arrive in a particular phase may not be transmitted in that phase.

Further, let N_k, be the number of packets at node TR at the beginning of phase k, and w_k be the window size allocated during phase k.

Define

$$S_k = (N_{k-1} - w_{k-1})^+, \tag{2.1}$$

where X^+ is the non-negative part of X.

S_k is the information available to RC at the beginning of phase k. This information is equivalent to N_{k-1}, since w_{k-1} is determined by RC in the previous phase. We shall term S_k to be the state of the system at the start of phase k.

We now define and motivate the objective function that we shall attempt to minimize. Recall that we have to balance the possible waste of buffers caused by over allocation with the possible excessive time-delays caused by under allocation. With this in mind, we define the cost incurred per phase, when there are n messages at node TR at the beginning of the phase and a window of size w is allocated to

$$b \cdot (w - n)^+ + h \cdot (n - w)^+ + c \cdot w,$$

where b, h and c are arbitrary non-negative constants.

Clearly, the values of the constants can be adjusted to reflect the operating environment of the system. For example, if buffer storage is heavily constrained, it may be desirable to have a large b and c but a small h. Conversely, if time delays are paramount, h can be set large and b and c small.

The problem of optimally controlling the flow between TR and RC is formulated as a Markov decision process.

Let π be any admissible control policy and $V_k^\beta(\pi,s)$ the total cost of using policy π and discount factor $0 < \beta \leq 1$, given that the system is at state s and there are k steps to go.
Also let

$$V^\beta(\pi,s) = \lim_{k\to\infty} V_k^\beta(\pi,s) \quad ; \quad V^\beta(s) = \inf_\pi V^\beta(\pi,s), \quad (\beta < 1),$$

and

$$\overline{V}(\pi,s) = \limsup_{k\to\infty} \frac{1}{k} V_k^1(\pi,s) \quad ; \quad \overline{V}(s) = \inf_\pi \overline{V}(\pi,s).$$

By an induction argument on k we show that

Lemma 1: For every $\beta < 1$, $V^\beta(s)$ is convex in s.

Consider the infinite horizon problem with discount factor $\beta < 1$, and let $w^*(s, \beta)$ be the optimal window size when the system is at state $S_k = s$.

From Lemma 1 we derive the optimal window size as a function of the state s.

Theorem 1: For every $\beta > 1 - \frac{h}{c}$, $w^*(s, \beta)$ is given by Figure 1, where $w^*(\beta) \geq 0$ is the optimal window size at state 0.

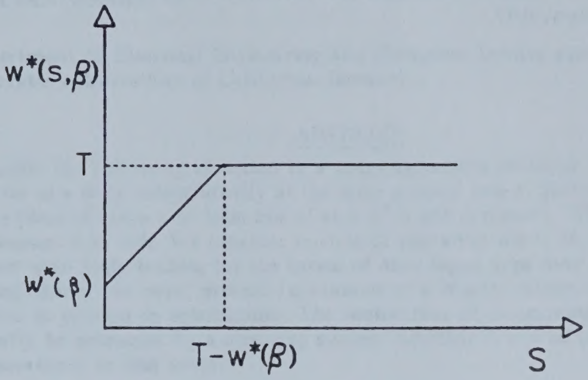


Figure 1: The optimal window size

It is seen that the optimal window size in this case is simply the current system state, s, added to the optimal window size that would have been allocated if the state had been zero, $w^*(\beta)$, truncated above by T. Thus, all that is necessary to compute the optimal control is to know the value of $w^*(\beta)$.

To obtain the optimal window size for the long-run average cost problem, we use the standard approach and let $\beta \rightarrow 1$. (See e.g. [Li].)

Theorem 2: The optimal window size in the long-run average cost problem is a stationary policy and has the form given in Figure 1 with $w^* = lim_{\beta \rightarrow 1} w^*(\beta)$.

The value w^* is found by evaluating $V^\beta(\pi,s)$ for policies π of the form given in Figure 1. This can be done since under such a policy, $V^\beta(\pi,s)$, $s = 0,1,2...$ have simple linear relations.

For example, when the arrival process is Bernoulli,

$$w^* = \min\left\{ w \mid B_T(w) > \frac{h}{h + c + b} \right\},$$

where $B_T(w) = Pr(Y \leq w)$ and Y is the sum of T independent Bernoulli r.v.'s with probability of success λ.

References

[Co] Cohen, J.W., "*The Single Server Queue,*" Northern-Holland, Amsterdam 1969.

[GeKl] Gerla, M. and Kleinrock, L,. "Flow Control Protocols", *Computer Network Architectures and Protocols,* (edited by P.E.Green, Jr.), Plenum Press, New York, pp. 361-412, 1982.

[Li] Lippman, S.A., "On Dynamic Programming with Unbounded Rewards," *Management Science,* 21, pp. 1225-1233, 1975.

[Ro] Ross, S.M., "*Applied Probability Models with Optimization Applications.*" Holden-Day, San Francisco, 1970.

[Sc] Schwartz, M., "*Computer Communication Networks, Design and Analysis.*" Prentice-Hall, Inc., New Jersey, 1977.

MODELLING THE FLOW OF COALESCING DATA STREAMS THROUGH A PROCESSOR

V. ANANTHARAM

Department of Electrical Engineering and Computer Science and Electronics Research Laboratory, University of California, Berkeley.

ABSTRACT

Consider the following situation in a communications network: Two data streams A and B arrive at a node independently at the same poisson rate λ. Service at exponential rate μ can take place iff there is at least one of each of A and B present. The output is the combined processed data AB. We consider models of this situation with finite buffers, infinite buffers and with finite buffers for the excess of each input type over the other. We apply the filtering theory for point process functionals of a Markov chain to study whether the output flow is poisson in equilibrium. The motivation is to examine, if the output is to subsequently be processed by a queueing system, whether it can be treated as an independent poisson input to that system.

Of independent interest, we prove that a subset of transitions of a countable state Markov process does not yield a poisson process when counted, if the rate matrix of counted transitions is nilpotent, and we prove a generalization of Pakes' lemma for countable state Markov chains.

1. INTRODUCTION

It is common practice to model arrival of customers into a network of service stations as a poisson process. Such models are extensively used to, e.g., understand the behaviour of data flow in a computer communications network. The servers in the network, (e.g. the CPU's of the various interconnected computers), are themselves modelled variously according to their service discipline. One of the remarkable discoveries in queueing theory, originating with the work of Burke [Bu], is that if the serving nodes are (well modelled as) quasireversible (see e.g. [K1], [W3]), then the output flows of the network are also poisson in equilibrium ([K2], [M], [WV]). This lends strength to the poisson approximation. It also accounts for the popularity of quasireversible nodes as models for processors.

Arguably the simplest model for a service facility with finite waiting room is the M/M/1/S FCFS queue. Customers arrive in a poisson stream of rate λ and require independent service times exponentially distributed with rate μ. Any customers arriving when the queue size is S are rejected.

It is known that the output of the M/M/1/S queue is not poisson in equilibrium.

If S is very large one may simplify the model further by allowing for arbitrary queue size, which gives the familiar M/M/1 FCFS model. This is a quasireversible node, and its output is therefore poisson in equilibrium.

There are situations in practice where different data streams have to be coalesced before processing. In a parallel processing system where portions of a job are worked on by different processors, a given processor may require inputs from more than one data stream before beginning processing. Alternately, to manufacture a commodity it may be necessary to have available different types of raw materials. We call a node modelling such a situation a **manufacturing** node. In this paper we investigate the possibility of coming up with a reasonable model of a manufacturing node with an equilibrium situation where poisson inputs yield a poisson output. The availability of such a model would greatly extend the range of data processing situations which can be modelled by quasireversible nodes.

The simplest model one could imagine for a manufacturing node with 2 inputs and finite waiting room is the following : Customers of types A and B arrive according to independent poisson streams of rate λ. Service at exponential rate μ can take place only if there is at least one of both A and B present. The output is the combined commodity AB. If an arrival of type A finds N_A customers of type A in queue on arrival, it is rejected. N_B is defined similarly. Call this the model with finite buffers.

In Section 3 we prove that the output in equilibrium of such a manufacturing node is not poisson.

If N_A or N_B is very large, one might try to simplify the model by ignoring the finite queue size limitation imposed by it. Call this the model with infinite buffers. (One or both of the buffers may be infinite). We may now ask if the output of the simplified model is poisson in equilibrium. It is natural to expect such a result by analogy with the M/M/1 queue.

Somewhat unexpectedly, it turns out that the question is meaningless, because there is no equilibrium situation. This is proved in Section 4. The point is that any excess of customers of one type over another tends to remain unchanged, whatever the rate of service. The behaviour of the model is similar to that of a null recurrent Markov chain.

As a final attempt to get a simple model, with the "infinite" features which make a poisson output plausible, we consider the following : Arrivals and service are as before. An arrival of type A is rejected on arrival if the **excess** of customers of type A over those of type B he sees on arrival is N_A. N_B is defined similarly. Call this the model with finite buffers for the excess.

The main result of this paper is that, even for this model, the output flow is not poisson.

In Section 5 we set up the Markov process associated to the model and determine a stationary range of the parameters. We prove a stationary distribution exists if $2\lambda < \mu$. The exact range of parameters permitting an equilibrium distribution is not determined. It appears to depend on N_A and N_B.

Calculations to prove that the output flow is not poisson in equilibrium are carried out in Section 6. Note that we avoid the problem of computing the invariant distribution of the chain, which would normally be required to study the filtering equations. Only its existence is used.

In Section 2 we summarize without proof the results we need from the filtering theory. The presentation here closely parallels Section 2 of [W1]. Brief concluding remarks are made in Section 7.

2. REVIEW OF THE FILTERING THEORY

Let $(X_t, t \geqslant 0)$ be a stationary Markov process, with countable state space X, infinitesimal generator Q and equilibrium distribution π. Let the process live on the probability space (Δ, G, P). Thus P is the law of the process.

Given a subset D of $\{ (i,j) \text{ s.t. } i,j \in X \text{ and } i \neq j \}$, define for $t \geqslant 0$.

$$D_t = \sum_{0 < s \leqslant t} 1 \{ (X_{s-}, X_s) \in D \} \tag{2.1}$$

$$D_0 = 0$$

The paths of $(D_t, t \geqslant 0)$ belong to Ω the collection of all functions $\omega : R_+ \rightarrow N$ of the form $\omega_t = \sum_{n \geqslant 1} 1 \{ t \geqslant T_n(\omega) \}$ where $\{T_n(\omega), n \geqslant 1\}$ is any sequence in $(0, \infty)$ strictly increasing to ∞ For $t \geqslant 0$, define $d_t(\omega) = \omega_t$ and $F_t = \sigma(d_s, 0 \leqslant s \leqslant t)$. Let $F = \bigvee_{t \geqslant 0} F_t$. The $T_n(\omega)$ are the jump times of $(d_t, t \geqslant 0)$. As a functional of the Markov process $(X_t, t \geqslant 0)$ with the law P, the process $(D_t, t \geqslant 0)$ induces a probability measure P_D on

(Ω, F).

To the law P_D and the filtration $(F_t, t \geqslant 0)$ on Ω there corrresponds a right continous, (F_t) -adapted process $(h_t^D(\omega), t \geqslant 0)$ having discontinuities only at the jump times of (d_t) and such that $d_t(\omega) - \int_0^t h_s^D(\omega)\, ds$ is a (P_D, F_t)- local martingale...... right continous version of the intensity. See [B] & [DM] for details.

It is known that $(D_t, t \geqslant 0)$ is poisson if and only if $h_t^D(\omega)$ is constant for almost all ω, for all $t \geqslant 0$, ([B], [DM]).

Define the matrices Q_D and Q^D by

$$Q_D\ (i,j) = Q\ (i,j)\, 1\, \{\, (i,j) \notin D\, \} \tag{2.2a}$$

and

$$Q^D\ (i,j) = Q\ (i,j)\, 1\, \{\, (i,j) \in D\, \} \tag{2.2b}$$

for each $i, j \in X$.

For $t \geqslant 0$, let ξ_t^D be the conditional distribution of X_t given $(D_s, 0 \leqslant s \leqslant t)$, i.e. a row vector with components

$$\xi_t^D(i) = P(X_t = i \mid (D_s, 0 \leqslant s \leqslant t)) \tag{2.3}$$

Then ξ_t^D is an (F_t) -adapted process on (Ω, F). It can be chosen right continous and so that it satisfies the following equations

$$\dot{\xi}_t^D = \xi_t^D Q_D + \xi_t^D h_t^D \qquad \text{for } t \notin \{T_n(\omega), n \geqslant 1\} \tag{2.4a}$$

$$\xi_t^D = \frac{\xi_{t-}^D Q^D}{h_{t-}^D} \qquad \text{for } t \in \{T_n(\omega), n \geqslant 1\} \tag{2.4b}$$

$$\xi_0^D = \pi \tag{2.4c}$$

where

$$h_t^D = \sum_{(i,j) \in D} \xi_t^D(i) Q(i,j) \tag{2.5}$$

and the derivative is right handed.

These **filtering equations** are a special case, in our context, of a much more general set of equations which can be explicitly written for the conditional expectation of a (F_t)-adapted semimartingale relative to an observation filtration (O_t), which is generated by a (F_t) -adapted multivariate point process (See [B]). On the other hand, there is a clear derivation of the above equations, exposing their intuitive content, due to Walrand [W2], which any reader who finds them mystifying is urged to study.

3. THE MODEL WITH FINITE BUFFERS

The Markov process associated to the model with finite buffers has the finite state space

$$X = \{\ (a, b)\ \text{s.t.}\ 0 \leqslant a \leqslant N_A,\ 0 \leqslant b \leqslant N_B\ \}$$

and the transition matrix Q can be read off from Fig. 1. It is irreducible, therefore admits a unique stationary distribution.

In the following theorem we give an easily verifiable necessary condition for the point process counting a subset of transitions of a countable state Markov process to be poisson.

3.1 Nilpotency Theorem :

Let $(X_t, t \geqslant 0)$ be a stationary Markov chain with countable state space X, infinitesimal generator Q, and invariant distribution π. Let D be a subset of transitions and $(d_t, t \geqslant 0)$ the associated point process, as in Section 2. If Q^D (Eqn. (2.2b)) is nilpotent then $(d_t, t \geqslant 0)$ is not poisson.

Proof : We first need to introduce the **unnormalized** version of the filtering equations, which are computationally easier to handle in the following proof. Let the random row vector valued function $\eta_t^D = (\eta_t^D(i), i \in X), t \geqslant 0$, evolve on the measure space (Ω, F, P_D) of Section 2 according to the equations

$$\dot{\eta}_t^D = \eta_t^D Q_D \qquad \text{for } t \notin \{T_n(\omega), n \geqslant 1\} \tag{3.1a}$$

$$\eta_t^D = \eta_{t-}^D Q^D \qquad \text{for } t \in \{T_n(\omega), n \geqslant 1\} \tag{3.1b}$$

$$\eta_0^D = \pi \tag{3.1c}$$

One verifies that $\eta_t^D, t \geqslant 0$ admits the interpretation of an unnormalized conditional density evolving so as to be adapted to the filtration F_t on (Ω, F) by checking that $\xi_t^D = \dfrac{\eta_t^D}{\eta_t^D \cdot 1}$ satisfies Eqns. (2.4a) -(2.4c), where 1 is a column vector of 1 's.

Equations (3.1a) -(3.1c) are the analog of the Duncan Mortensen Zakai equations in the filtering theory for semimartingales of a Wiener process.

Suppose that $(d_t, t \geqslant 0)$ is poisson with intensity λ. Then, for almost all $\omega \in \Omega$ we have, for all $t \geqslant 0$

$$\eta_t^D Q_D 1 = -\lambda \eta_t^D 1 \tag{3.2}$$

Differentiating this in a neighbourhood of $t = 0$ and applying Eqn (3.), gives

$$\eta_t^D (Q_D)^k 1 = (-\lambda)^k \eta_t^D 1 \tag{3.3}$$

which holds, for a.a. ω, in a neighbourhood of $t = 0$ (depending on ω), for each $k \geqslant 0$. We have used that $T_1(\omega) > 0$ $a.s.$ (P_D).

In particular, setting $t = 0$ in the above yields

$$\pi (Q_D)^k 1 = (-\lambda)^k \tag{3.4}$$

for all $k \geqslant 0$.

We remark that as a consequence of the above, $-\lambda$ must necessarily be an eigenvalue of Q_D when X is finite.

Now suppose that, instead of counting every transition the Markov process makes in D, we choose $0 \leqslant p \leqslant 1$ and make, at each transition, an independent (of the past of $(X_t, t \geqslant 0)$ at the instant of choice) decision of counting the transition with probability p. Thus we get a point process functional $(D_t^p, t \geqslant 0)$ of the Markov process, which is a Bernoulli sampled version of $(D_t, t \geqslant 0)$, and takes values in Ω where it induces a measure, and where the process of evaluations $(d_t, t \geqslant 0)$ induces the filtration F_t as in section 2. For clarity we will write Ω_p for Ω with this measure and $(d_t^p, t \geqslant 0)$ for $(d_t, t \geqslant 0)$ in this context. Then filtering equations may be written exactly as before for the conditional distribution of the Markov process given the Bernoulli sampled observations. An unnormalized version is given by the random row vector valued function η_t^{Dp} evolving on Ω_p according to the equations

$$\dot{\eta}_t^{Dp} = \eta_t^{Dp} (Q_D + q Q^D) \qquad \text{for } t \notin \{T_n(\omega), n \geqslant 1\} \tag{3.5a}$$

$$\eta_t^{Dp} = \eta_{t-}^{Dp} p Q^D \qquad \text{for } t \in \{T_n(\omega), n \geqslant 1\} \tag{3.5b}$$

$$\eta_t^{Dp} = \pi \tag{3.5c}$$

where $q = 1 - p$ as usual.

Recall that a p -Bernoulli sampling of a poisson process with intensity λ yields a poisson process of intensity λp. Therefore $(d_t^p, t \geqslant 0)$ is poisson with intensity λp, and when one reasons as was done above with $p = 1$, one gets

$$\eta_t^{D,p} (Q_D + q\, Q^D)^k\, 1 = (-\lambda p)^k\, \eta_t^{D,p}\, 1 \tag{3.6}$$

holding for a.a. ω, in a neighbourhood of $t = 0$ for all $k \geqslant 0$.

Setting $t = 0$ gives

$$\pi (Q_D + q\, Q^D)^k\, 1 = (-\lambda p)^k \tag{3.7}$$

Let us now think of Eqn (3.7) as a family of equations parametrized by p and differentiate k times with respect to p. It is easily checked that this gives

$$\pi (Q^D)^k\, 1 = \lambda^k \tag{3.8}$$

for all $k \geqslant 0$.

From Eqn (3.8) we conclude

a) When X is finite λ is an eigenvalue of Q^D. In fact if Min (x) denotes the minimal polynomial of Q^D, so that Min $(Q^D)= 0$, and Min (x) is the polynomial of least degree achieving this, one checks from Eqn (3.8) that Min $(\lambda)= 0$.

b) If Q^D is nilpotent, i.e. some power of it is 0, the point process associated to D cannot be poisson. □

To apply the above to the output of the model with finite buffers note that the subset D of transitions which gives the output process has an associated Q^D which is nilpotent (See Fig. 1). Thus the output process is not poisson.

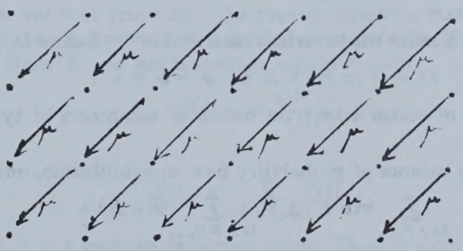


FIG 1

Aside : One also sees, as an application of the nilpotency theorem, that neither process of accepted arrivals is poisson, since they are point process functionals corresponding to the sets of transitions

$$A = \{ ((a,b), (a+1,b)) \text{ s.t. } 0 \leqslant a \leqslant N_A - 1, 0 \leqslant b \leqslant N_B \}$$

for process of accepted arrivals of type A, and

$$B = \{ ((a,b), (a,b+1)) \text{ s.t. } 0 \leqslant a \leqslant N_A, 0 \leqslant b \leqslant N_B - 1 \}$$

for type B, and the nilpotency of the appropriate matrices may be read off from Fig. 1.

Remark : The nilpotency theorem may be applied to make a number of similar statements about the flows of various queues.

4. THE MODEL WITH INFINITE BUFFERS

In this section we will study the model of Section 3. with either one or both of N_A and N_B infinite. For the sake of definiteness assume that N_A is infinite. Also assume $1 \leqslant N_B \leqslant \infty$ to avoid triviality.

We show below that the Markov process associated to this model does not admit a stationary distribution.

The state space associated to the model is

$$X = \{ (a,b) \text{ s.t. } 0 \leqslant a < \infty, 0 \leqslant b \leqslant N_B \}$$

and the infinitesimal generator Q can be read off from Fig. 2.

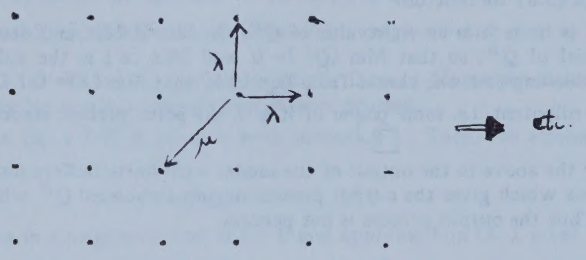


FIG. 2

Suppose the process admits the invariant distribution π. Define $D_n \subseteq X$, $n \geqslant 0$, by

$$D_n = \{ (a,b) \in X \text{ s.t. } a - b = n \}$$

so that D_n is the subset of states where the excess of customers of type A over those of type B is n .

From Fig. 2. and the balance of probability flow in equilibrium, one easily sees that

$$\sum_{(a,b) \in D_n} \pi(a,b)\,\lambda \leq \sum_{(a,b) \in D_{n+1}} \pi(a,b)\,\lambda \tag{4.1}$$

for each $n \geqslant 0$.

Since $\sum_{(a,b) \in X} \pi(a,b) = 1$, this can only hold if

$$\pi(a,b) = 0 \tag{4.2}$$

for each $(a,b) \in \bigcup_{n \geqslant 0} D_n$.

It is well known that if an irreducible process admits an invariant distribution, the distribution is strictly positive on the state space. Our process is clearly irreducible, so that the above contradiction proves it cannot admit an invariant distribution.

Note that the size of $\frac{\mu}{\lambda}$ was completely irrelevant to the above argument.

5. THE MODEL WITH FINITE BUFFERS FOR THE EXCESS

The state space X of the model with finite buffers for the excess is

$$X = \{ (a,b) \text{ s.t. } a,b \geqslant 0,\ a-b \leqslant N_A \text{ and } b-a \leqslant N_B \} \tag{5.1}$$

The off diagonal entries of the infinitesimal generator are the rates of the various transitions and are given by

$$Q(\,(a,b),(a+1,b)\,) = \lambda \cdot 1\{ a-b \neq N_A \} \tag{5.2a}$$

$$Q(\,(a,b),(a,b+1)\,) = \lambda \cdot 1\{ b-a \neq N_B \} \tag{5.2b}$$

$$Q(\,(a,b),(a-1,b-1)\,) = \mu \cdot 1\{ a > 0 \} \cdot 1\{ b > 0 \} \tag{5.2c}$$

See Fig. 3 .

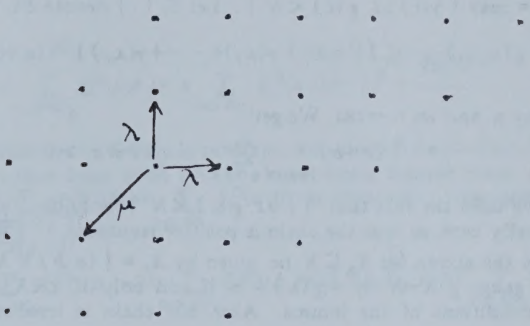


FIG. 3

In this section we will prove that the process admits a stationary distribution when $2\lambda < \mu$. To do this we consider the imbedded discrete time Markov chain (x_n), which has the same state space X, and has transition matrix Γ with

$$\Gamma(i,j) = \frac{Q(i,j)}{Q(i)} \tag{5.3}$$

where

$$Q(i) = \sum_{j \in X,\, j \neq i} Q(i,j) \tag{5.4}$$

and prove that (x_n) is a positive recurrent chain. Thus it admits an invariant distribution. Since the original Markov process is stable and conservative, the existence of an invariant distribution for the process is equivalent to the existence of an invariant distribution for the imbedded chain.

We need the following generalization of Pakes' Lemma ,whose ordinary version, applicable to chains with state space the nonnegative inttegers, is proved in e.g. [W3] .

Generalized Pakes' Lemma :

Let $(x_n, n \geqslant 0)$ be an irreducible Markov chain, with state space X. Let there be given a function $g: X \to N$, such that, for each $N > 0$, $\{ x \in X \text{ s.t. } g(x) \leqslant N \}$ is finite.

For $i \in X$, define

$$\gamma(i) = E(\, g(x_{n+1}) - g(x_n) \mid x_n = i \,) \tag{5.5}$$

Suppose

a) $|\gamma(i)| < \infty,\ \forall\, i \in X$.

b) There is $N > 0$ and $-a < 0$ such that for each $i \in X$ with $g(i) \geqslant N$, we have $\gamma(i) \leqslant -a$.

Then $(x_n, n \geqslant 0)$ is a positive recurrent chain. In particular it admits an invariant distribution.

Remark : We take as known, that for an irreducible Markov chain of period d, and $i, j \in X$

$$\lim_{n \to \infty} \frac{\sum_n P(x_n = j \mid x_0 = i)}{n} = \frac{\pi(j)}{d}$$

exists. Also, $\pi(j) > 0$ if and only if the chain is positive recurrent, admitting invariant distribution $\pi(.)$, (and equals 0 otherwise).

Proof : Let $b = \max\{\gamma(i) \ s.t. \ g(i) < N\}$. Let $E_i\{\,.\,\}$ denote $E(.\mid x_0 = i)$. We have

$$-g(i) \leqslant E_i\{g(x_{n+1}) - g(x_0)\} = E_i\{\gamma(x_1) + \cdots + \gamma(x_n)\} \leqslant (a+b) \sum_{m=0}^{n} P_i(g(x_m) < N) - na$$

Divide by n and let $n \to \infty$ We get

$$(a+b) \sum_{\{i \ s.t. \ g(i) < N\}} \pi(i) \geqslant a > 0$$

where we have used the fact that $\{i \ s.t. \ g(i) < N\}$ is finite. This proves that $\pi(.)$ cannot be identically zero, so that the chain is positive recurrent. □

To apply the above, let $X_n \subseteq X$ be given by $X_n = \{(a, b) \in X \ s.t. \ \min\{a, b\} = n\}$

Define the gauge $g: X \to N$ by : $g(x) = n$ if and only if $x \in X_n$. If $2\lambda < \mu$ then $g(.)$ satisfies the conditions of the lemma. Also, our chain is irreducible and periodic with period 3 .

6. INTENSITY CALCULATIONS

To begin, observe from Fig. 1. that for each $n \geqslant 0$

$$\lambda \sum_{i \in X_n} \xi_t^D(i) > \mu \sum_{i \in X_{n+1}} \xi_t^D(i) \tag{6.1}$$

Summing over all n gives

$$\lambda > h_0^D \tag{6.2}$$

where we have used that

$$h_t^D = \mu \sum_{(a,b) \in X} \xi_t^D(a,b) 1\{a > 0\} 1\{b > 0\} \tag{6.3}$$

and

$$\xi_0^D(a,b) = \pi(a,b) \tag{6.4}$$

This will be used below.

In the following all sums with unspecified range are over the entire state space. Our calculations will all be done in a neighbourhood of $t = 0$. Thus we will not need to use the jump part of the filtering equations. Written out explicitly, the smooth part of the equations read

$$\begin{aligned} \dot{\xi}_t^D(a,b) = {} & \lambda \xi_t^D(a-1,b) 1\{a > 0\} 1\{b - a < N_B\} \\ & + \lambda \xi_t^D(a, b-1) 1\{b > 0\} 1\{a - b < N_A\} \\ & - \xi_t^D(a,b)(\lambda 1\{a - b < N_A\} + \lambda 1\{b - a < N_B\} + \mu 1\{a > 0\} 1\{b > 0\}) \\ & + \xi_t^D(a,b) h_t^D \end{aligned} \tag{6.5}$$

Sum over $\{ (a,b) \text{ s.t. } a,b > 0 \}$ to get

$$\frac{\dot{h}_t^D}{\mu} = \lambda \sum_{0<b\leq N_B} \xi_t^D(0,b) + \lambda \sum \xi_t^D(a,b) 1\{a>0\} 1\{b>0\} 1\{a-b<N_A\}$$

$$+ \lambda \sum_{0<a\leq N_A} \xi_t^D(a,0) + \lambda \sum \xi_t^D(a,b) 1\{a>0\} 1\{b>0\} 1\{b-a<N_B\}$$

$$- \sum \xi_t^D(a,b) 1\{a>0\} 1\{b>0\} (\lambda 1\{a-b<N_A\} + \lambda 1\{b-a<N_B\}$$

$$+ \mu 1\{a>0\} 1\{b>0\}) + \frac{(h_t^D)^2}{\mu} \tag{6.6}$$

Simplyfying

$$\frac{\dot{h}_t^D}{\mu} = \lambda \sum_{0<b\leq N_B} \xi_t^D(0,b) + \lambda \sum_{0<a\leq N_A} \xi_t^D(a,0) - h_t^D + \frac{(h_t^D)^2}{\mu} \tag{6.7}$$

Since we have assumed the intensity is constant, and since $T_1(\omega) > 0$ *a.s.* (P_D), we see from the above equation that there must be, for almost every sample point, a neighbourhood of $t = 0$ where $\sum_{0<b\leq N_B} \xi_t^D(0,b) + \sum_{0<a\leq N_A} \xi_t^D(a,0)$ is constant. Now compute, away from the jump points, if $N_B > 0$

$$\sum_{0<b\leq N_B} \dot{\xi}_t^D(0,b) = \lambda \sum_{0\leq b<N_B} \xi_t^D(0,b) - 2\lambda \sum_{0<b<N_B} \xi_t^D(0,b) - \lambda \xi_t^D(0,N_B) + h_t^D \sum_{0<b\leq N_B} \xi_t^D(0,b)$$

Simplyfying

$$\sum_{0<b\leq N_B} \dot{\xi}_t^D(0,b) = \lambda \xi_t^D(0,0) + (h_t^D - \lambda) \sum_{0<b\leq N_B} \xi_t^D(0,b) \tag{6.8a}$$

Similarly, we get, away from the jumps, if $N_A > 0$

$$\sum_{0<a\leq N_A} \dot{\xi}_t^D(a,0) = \lambda \xi_t^D(0,0) + (h_t^D - \lambda) \sum_{0<a\leq N_A} \xi_t^D(a,0) \tag{6.8b}$$

Summing equations (6.8a) and (6.8b), or taking them individually, as appropriate, we see that there must be, for almost every sample point, a neighbourhood of $t = 0$ where $\xi_t^D(0,0)$ is constant. (Note that at least one of N_A and N_B must exceed 0 to avoid triviality).

However

$$\dot{\xi}_t^D(0,0) = (h_t^D - 2\lambda)\, \xi_t^D(0,0) \tag{6.9a}$$

if $N_A > 0$ and $N_B > 0$, and

$$\dot{\xi}_t^D(0,0) = (h_t^D - \lambda)\, \xi_t^D(0,0) \tag{6.9b}$$

if one of N_A and N_B is zero.

In either case, since $h_0^D < \lambda$ there can be no neighbourhood of $t = 0$ where $\xi_t^D(0,0)$ is constant.

This contradiction establishes that the output flow of the manufacturing system cannot be poisson in equilibrium.

7. CONCLUDING REMARKS

In the preceding we have considered several attempts at defining a manufacturing node which admits a stationary situation yielding a poisson output for poisson inputs.

Our results are negative, in that several natural definitions are shown to not possess the property we seek. Perhaps there is no such model (?). There has been some recent work on measuring the distance of a process from poisson-ness in terms of the intensity [BP]. An interesting direction to go from here would be to quantify the deviation from poissonness of the outputs of the above, and of other models.

References

[B] Bremaud, P. Point processes and Queues. Springer Verlag 1981.

[Bu] Burke, P.J. The output of a queueing system. Op. Res. 4, 699 -704 (1956).

[BP] Brown, T.C. and Pollett, P.K. Some distributional approximations in Markovian queueing networks. Adv. Appl. Prob. 14, 654 -671 (1982).

[DM] Claude Dellacherie and Paul Meyer. Probabilities and Potential. North Holland. (1975).

[K1] Kelly, F. Reversibility and Stochastic Networks. Wiley, London. (1979).

[K2] Kelly, F. Networks of queues with customers of different types. J. App. Prob. 12, 542 -554 (1975).

[M] Melamed, B. On poisson transition processes in discrete state Markovian systems with applications to queueing theory. Adv. Appl. Prob. 11, 218 -239 (1979).

[W1] Walrand, J. On the equivalence of flows in networks of queues. J. App. Prob. 19, 195 -203 (1982).

[W2] Walrand, J.. Filtering formulas and the ./M/1 queue in a quasireversible network. Stochastics 6, 1 -22 (1981).

[W3] Walrand, J.. Lectures on Queueing Theory. (EECS 228 at U.C. Berkeley) (1985). In preparation.

[WV] Walrand, J. and Varaiya, P. Flows in Queueing Networks : A Martingale approach. Mathematics of Operations Research 6, No. 3 (1981).

Calculating Exit Times for Series Jackson Networks I.

William A. Massey

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

We define a new family of special functions that we call lattice Bessel functions. They are indexed by the N-dimensional integer lattice such that they reduce to modified Bessel functions when $N = 1$, and the exponential function when $N = 0$. The transition probabilities for an $M/M/1$ queue going from one state to another before becoming idle (exiting at zero) can be solved in terms of modified Bessel functions. In this paper, we use lattice Bessel functions to solve the analogous problem involving the exit time from the interior of the positive orthant of the N-dimensional lattice for a series Jackson network with N nodes. These special functions allow us to derive asymptotic expansions for the taboo transition probabilities, as well as for the tail of the exit time distribution.

Calculating Exit Times for Series Jackson Networks I.

William A. Massey

AT&T Bell Laboratories
Murray Hill, New Jersey 07974

1. INTRODUCTION

Consider the queue length process for the **M/M/1** system with Poisson arrival rate λ and exponential service rate μ. We can think of it as the reflecting version of the following Markov process. Let $Z(t) = N_\lambda(t) - N_\mu(t) + n$, where $N_\lambda(t)$ and $N_\mu(t)$ are independent Poisson processes with intensities λ and μ respectively. With $N_\lambda(0) = N_\mu(0) = 0$, we have $Z(0) = n$ where n belongs to the set of integers Z. This process is sometimes called a randomized random walk on Z (see Feller [1], page 59).

If n belongs to Z_+, the set of non-negative integers, then $Z(t)$ behaves the same as the **M/M/1** queue length process until it hits the zero state. From here, $Z(t)$ may take on negative values, so its state space will be all of Z. Given this relationship, we can think of the busy period for an **M/M/1** queue as being the time for an identically initialized $Z(t)$ process to be absorbed at the zero state. Let $p_t(m,n)$ be the transition probability for $Z(t)$ starting at m and terminating at n, for time $t > 0$ and all m and n in Z. For m and n in Z_+, we let $q_t(m,n)$ be a similar transition probability from m to n with the additional requirement that $Z(s)$ does not equal zero for $0 \leqslant s \leqslant t$. It is well known (see Ledermann and Reuter [2]) that $p_t(m,n)$ and $q_t(m,n)$ can be solved explicitly in terms of modified Bessel functions as

$$p_t(m,n) = e^{-(\lambda+\mu)t} \left[\frac{\lambda}{\mu} \right]^{\frac{n-m}{2}} I_{n-m}(2t\sqrt{\lambda\mu})$$

$$q_t(m,n) = e^{-(\lambda+\mu)t} \left[\frac{\lambda}{\mu} \right]^{\frac{n-m}{2}} \left[I_{n-m}(2t\sqrt{\lambda\mu}) - I_{n+m}(2t\sqrt{\lambda\mu}) \right].$$

In this paper, we shall generalize these two results. Just as these formulas describe processes associated with the **M/M/1** queue length process, we shall construct analogous processes for the N-node series (or pipeline) Jackson network. In particular, the analogue to $q_t(m,n)$ will be the transition probabilities for such a network before *any* one of the servers becomes idle.

To generalize the solutions to $p_t(m,n)$ and $q_t(m,n)$, we need to define a new class of functions that generalize the notion of a modified Bessel function. Recall the following formula for modified Bessel functions

$$e^{\frac{y}{2}[x+\frac{1}{x}]} = \sum_{n\in Z} x^n I_n(y).$$

For $\mathbf{n} = (n_1,\ldots,n_N)\in Z^N$ we say that $I(\mathbf{n},y)$ is a *lattice-Bessel function of rank N* if it is defined by the following generating function relation:

$$\exp\left(\frac{y}{N+1}\left[x_1+\frac{x_2}{x_1}+\ldots+\frac{x_N}{x_{N-1}}+\frac{1}{x_N}\right]\right) = \sum_{\mathbf{n}\in Z^N} x_1^{n_1}\ldots x_N^{n_N} I(\mathbf{n},y).$$

From this defining relation follows various properties of lattice-Bessel functions that we will use in Section 2 and prove in Section 3:

[Representations for $I(\mathbf{n},y)$]

$$\frac{1}{(2\pi)^N}\int_{-\pi}^{\pi}\ldots\int_{-\pi}^{\pi}\exp\left[\frac{y}{N+1}(e^{i\theta_1}+e^{i(\theta_2-\theta_1)}+\ldots+e^{i(\theta_N-\theta_{N-1})}+e^{-i\theta_N})-i\sum_{j=1}^{N}n_j\theta_j\right]d\theta_1\ldots d\theta_N \quad (1.1)$$

$$\sum_{j=0}^{\infty}\prod_{k=0}^{N}\frac{\left(\frac{y}{N+1}\right)^{j+\sum_{\ell=k+1}^{N}n_\ell}}{\left(j+\sum_{\ell=k+1}^{N}n_\ell\right)!} \quad (1.2)$$

[Symmetry group]

Let G_N be the set of matrices that permute the set of vectors $\{\mathbf{e}_1,\mathbf{e}_2-\mathbf{e}_1,\ldots,\mathbf{e}_N-\mathbf{e}_{N-1},-\mathbf{e}_N\}$, where $\mathbf{e}_k$ is the kth unit basis vector of Z^N. We then have G_N isomorphic to S_{N+1}, the group of permutations on $N+1$ objects. For all $\boldsymbol{\pi}$ in $Aut(Z^N)$ we have $I(\mathbf{n},\cdot) = I(\boldsymbol{\pi}(\mathbf{n}),\cdot)$ for all $\mathbf{n}$ in Z^N, if and only if $\boldsymbol{\pi}$ belongs to G_N. Consequently, G_N is the symmetry group for the rank N lattice-Bessel functions.

[Asymptotics]

$$I(\mathbf{n},y) = \frac{e^y}{\sqrt{N+1}}\left(\frac{N+1}{2\pi y}\right)^{N/2}\left[1+\left(\frac{N(N+2)}{24}-\frac{1}{2}\mathbf{B}(\mathbf{n}^*)\right)\frac{1}{y}+0\left(\frac{1}{y^2}\right)\right]$$

where $\mathbf{n}^* = (n_1^*,\dots,n_N^*)$ is derived from $\mathbf{n}$ by setting $n_j^* = \sum_{k=j}^{N} n_k$, and $\mathbf{B}(\mathbf{n}^*)$ equals the quadratic form $\sum_{j,k=1}^{N} b_{jk}\, n_j^* n_k^*$, with $b_{jj} = N$ for all j, and $b_{jk} = -1$ for $j \neq k$.

Consider an N-node series Jackson network with Poisson input rate λ and service rate μ_i for the $i-th$ exponential server. Our analogue to $Z(t)$ will be such that the above process is a reflecting version of it. If $N_\lambda(t)$, $N_{\mu_1}(t)$, $\cdots$, $N_{\mu_N}(t)$ are $N+1$ independent Poisson processes, and $\mathbf{Z}(0) = (n_1,\dots,n_N)$, we define $\mathbf{Z}(t)$ to be

$$\left(N_\lambda(t) - N_{\mu_1}(t) + n_1,\ N_{\mu_1}(t) - N_{\mu_2}(t) + n_2,\ \dots,\ N_{\mu_{N-1}}(t) - N_{\mu_N}(t) + n_N\right).$$

Let α and γ be respectively the arithmetic and geometric means of λ, $\mu_1,\dots,\mu_N$ or

$$\alpha = \frac{\lambda + \sum_{j=1}^{N} \mu_j}{N+1} \qquad \gamma = \left[\lambda \cdot \prod_{j=1}^{N} \mu_j\right]^{\frac{1}{N+1}}.$$

Now define $\beta_1,\dots,\beta_N$ so that

$$\lambda = \beta_1 \gamma\,,\ \mu_1 = \frac{\beta_2}{\beta_1}\gamma, \dots, \mu_{N-1} = \frac{\beta_N}{\beta_{N-1}}\gamma\,,\ \mu_N = \frac{1}{\beta_N}\gamma.$$

We then get $\beta_1 = \dfrac{\lambda}{\gamma}$ and $\beta_j = \dfrac{\lambda\mu_1\cdots\mu_{j-1}}{\gamma^j}$ for $j = 2,\dots,N$. For $\mathbf{n} \in Z^N$, let $\boldsymbol{\beta}^{\mathbf{n}} = \prod_{i=1}^{N} \beta_i^{n_i}$. We can now write the transient behavior for $\mathbf{Z}(t)$ compactly.

Theorem 1.1. *If* $p_t(\mathbf{m},\mathbf{n}) = Pr\{Z(t) = \mathbf{n} | Z(0) = \mathbf{m}\}$ *for all* $\mathbf{m}$ *and* $\mathbf{n}$ *in* Z^N, *then*

$$p_t(\mathbf{m},\mathbf{n}) = e^{-(N+1)\alpha t} \boldsymbol{\beta}^{\mathbf{n}-\mathbf{m}} \cdot I(\mathbf{n}-\mathbf{m},(N+1)\gamma t).$$

If $\mathbf{Z}(0)$ equals an element in the interior of Z_+^N, where each component is strictly positive, then let T equal the first time that $\mathbf{Z}(t)$ has a zero component. We then call T a random stopping time. The joint distribution of $\mathbf{Z}(t)$ with the event $\{T > t\}$ represents a new Markov process which acts like $\mathbf{Z}(t)$ but is absorbed when it touches the boundary of Z_+^N. In queueing theoretic terms, T equals the time until one of the queues becomes empty in a Jackson series network.

Theorem 1.2. *If* $q_t(\mathbf{m},\mathbf{n}) = Pr\{Z(t) = \mathbf{n},\ T > t \,|\, Z(0) = \mathbf{m}\}$ *for all* $\mathbf{m}$ *and* $\mathbf{n}$ *in* Z_+^N, *then*

$$q_t(\mathbf{m},\mathbf{n}) = e^{-(N+1)\alpha t} \boldsymbol{\beta}^{\mathbf{n}-\mathbf{m}} \sum_{\boldsymbol{\pi} \in G_N} (-1)^{\boldsymbol{\pi}} I(\mathbf{n}-\boldsymbol{\pi}(\mathbf{m}),(N+1)\gamma t). \tag{1.3}$$

where $(-1)^{\boldsymbol{\pi}}$ *is the sign of* $\boldsymbol{\pi}$, *viewed as a permutation.*

Given $p_t(\mathbf{m},\mathbf{n})$ and $q_t(\mathbf{m},\mathbf{n})$ in terms of lattice Bessel functions, we can now derive asymptotic formulas for them,

Theorem 1.3. *For* $t \uparrow \infty$, *we have*

$$p_t(\mathbf{m},\mathbf{n}) = \boldsymbol{\beta}^{\mathbf{n}-\mathbf{m}} \frac{e^{-(N+1)(\alpha-\gamma)t}}{\sqrt{N+1}\,(2\pi\gamma t)^{N/2}} \left[1 + \left(\frac{N(N+2)}{24(N+1)\gamma} - \frac{\mathbf{B}(\mathbf{n}^* - \mathbf{m}^*)}{2(N+1)\gamma} \right) \frac{1}{t} + O\left(\frac{1}{t^2} \right) \right]$$

$$q_t(\mathbf{m},\mathbf{n}) = -\boldsymbol{\beta}^{\mathbf{n}-\mathbf{m}} \frac{e^{-(N+1)(\alpha-\gamma)t}}{2(N+1)^{3/2}\gamma t\,(2\pi\gamma t)^{N/2}} \cdot \sum_{\pi \in G_N} (-1)^{\boldsymbol{\pi}} \mathbf{B}(\mathbf{n}^* - \boldsymbol{\pi}(\mathbf{m})^*) \cdot \left[1 + 0\left(\frac{1}{t} \right) \right]$$

Moreover, if $\beta_i < 1$ *for all* i, *then*

$$Pr\{T > t \,|\, \mathbf{Z}(0) = \mathbf{m}\} = O\left(\frac{e^{-(N+1)(\alpha-\gamma)t}}{t\sqrt{t^N}} \right).$$

BIBLIOGRAPHY

[1] Feller, William. *An Introduction to Probability Theory and its Applications.* Volume II. John Wiley and Sons, Inc. 1971.

[2] Ledermann, W. and Reuter, G. E. H. *Spectral Theory for the Differential Equations of Simple Birth and Death Processes*, Phil. Trans. Roy. Soc. A, 246, pp. 321-369, 1954.

[3] Massey, W. A., *An Operator Analytic Approach to the Jackson Network.* Journal of Applied Probability, 21, 379-393 (1984).

[4] Massey, W. A. and Wright, P. E., *Asymptotic Expansions for Lattice Bessel Functions* (in preparation).

ANOTHER LOOK AT INSENSITIVITY

JEAN WALRAND
Department of Electrical Engineering and Computer Sciences
and Electronics Research Laboratory
University of California, Berkeley CA 94720

ABSTRACT

Insensitivity is a property of some stochastic processes. In the Generalized Semi-Markov Process (GSMP) formulation, insensitivity refers to the fact that the invariant distribution of the *state* of the process depends on the lifetime distributions of some of its *clocks* only through their mean values.

This paper shows that the insensitivity is particular form of product form result. This explains the similarity between the product form and the insensitivity conditions found in the literature, and this permits simple proofs of insensitivity results.

1. INTRODUCTION

We will discuss insensitivity [2]-[4] in the context of GSMPs [3] (see section 2) and show that it can be seen as a product form result [1]. The discussion can be extended to the GSMPs with speeds [4]. The idea is to consider the GSMP as consisting of interacting Markov chains: the state and the clocks. A sufficient condition for insensitivity is then that these Markov chains be quasi-reversible with respect to the transitions causing the interactions.

2. GENERALIZED SEMI-MARKOV PROCESSES

The GSMPs are a generalization of the semi-Markov processes (SMPs). In a SMP, a clock is running and a state x_t (taking values in a countable set X) makes a random transition when the clock runs out: a new initial value of the clock (with a distribution that depends on the value of the state) is then chosen and the evolution continues.

A GSMP is similar except that there is a set of running clocks. This set is decomposed into exponentially and nonexponentially distributed clocks.

To simplify the subsequent discussion, we will consider that x_t is a regular Markov chain on some X with a life of its own: thus all the exponential clocks are taken into account in the rate matrix Q of x_t. Notice that one can also have some nonexponential clocks that were given a phase type representation in x_t. In addition, there is a set J of distinguished nonexponential clocks. By a clock, we will always mean one of those distinguished clocks.

A transition of x_t may start a new clock $i \in J$. Each clock i has a given distribution $F_i(t)$ for its initial value. In addition to its Markov chain transitions, the state x_t jumps when the first clock runs out (with a distribution that depends on which clock that is); and the evolution continues with each "old" clock continuing its count down. It is assumed that the value of x_t specifies which clocks are running.

The Markov description of a GSMP involves $z_t = (x_t, c_t^i, i \in J)$ where x_t is the state taking values in a countable set X and the c_t^i are the residual values of the clocks. The term *state* is of course a misnomer for x_t since it is not Markov in general.

Assume that the Markov process z_t admits an invariant distribution $\phi(z)$. Denote by $\pi(x)$ the corresponding marginal distribution of x in X. The insensitivity property refers to the fact that π depends on the F_i only through their mean values.

Insensitivity, in the queueing literature, occurs when ϕ has the form

$$\phi(z) = \pi(x) \prod_{i \in J(x)} R_i(t_i) \text{ for } z = (x, t_i, i \in J) \qquad (1)$$

where π is insensitive, $J(x)$ is the set of clocks that are running in state z, and $R_i(t) = \lambda_i \int_0^t [1 - F_i(s)] ds$, where λ_i^{-1} is the mean value of F_i. (In (1), $\phi(z) = Prob\{x_t = x, c_t^i \leq t_i, i \in J\}$.)

The insensitivity result is as follows [2]-[3].

Theorem

Assume that

$$\sum_y \pi(y) q(y, x) 1\{y \in D_i(x)\} = \lambda_i \pi(x), \; x \in X, \; i \in J(x) \qquad (2)$$

where π is invariant for x_t when the clocks are all exponentially distributed with respective means λ_i^{-1}, and where $D_i(x)$ is the set of y such that a transition from y to x starts clock i.

Then (1) is invariant for z_t.

3. PRODUCT FORM INTERPRETATION

Consider x_t as a queue used by customers of classes $i \in J$. Consider the set of clocks as being a M/G/∞ queue. Clock i is started when a customer of class i enters the M/G/∞ queue; it runs out when the customer leaves the queue. Consider the GSMP as being a closed network with two queues: the queue x_t and the M/G/∞ queue. The closed network has one customer of class i for $i \in J$.

Assume that the conditions of the Theorem hold.

Extend (fictitiously) queue x_t so that it admits Poisson arrivals of rate λ_i of class $i \in J$. This is done by making an arriving customer of class i jump the queue when there already is such a customer in the queue. It is then easy to see that (2) implies that x_t is a quasi-reversible queue. The M/G/∞ queue is trivially quasi-reversible. As a consequence, the closed network is product form. A moment of reflection shows that its invariant distribution on the set with one customer of class i for $i \in J$ is (1).

4. ACKNOWLEDGEMENTS

This work, supported in part by N.S.F. Grant No. ECS-8421128, was motivated by enjoyable discussions with Prof. Foley and Schassberger in Oberwolfach, and with Mr. Parekh and Tsoucas in Berkeley.

5. REFERENCES

[1] F.P. Kelly (1979). *Reversibility and stochastic networks*. Wiley.

[2] K. Matthes (1962), "Zur Theorie der Bedienungsprozesses," *Trans. 3rd Prague Conf. Information Theory*.

[3] R. Schassberger (1978), "Insensitivity of steady state distributions of generalized semi-Markov processes - Part II," *Ann. Prob.* **6,** 85-93.

[4] R. Schassberger (1978), "Insensitivity of steady-state distributions of generalized semi-Markov processes with speeds," *Adv. Appl. Prob.* **10,** 836-851.

THE NUMERICAL SOLUTION OF LINEAR DIFFERENTIAL EQUATIONS ON PARALLEL PROCESSORS BY ASYNCHRONOUS RELAXATIONS

DEBASIS MITRA

AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, New Jersey 07974

ABSTRACT

We consider chaotic, asynchronous relaxations for the numerical solution by parallel processors of initial value problems involving a class of ordinary differential equations. The differential equations may be non-linear, non-autonomous and non-homogeneous; however, the analytic results in this paper are for linear systems. The proposed algorithm has several advantages in the parallel processing context, and these apply irrespective of the number of processors and the architecture of the computing system. The main attraction of the algorithm is that it is essentially asynchronous, requiring negligible coordination between tasks, and the idle times of processors are negligible.

1. INTRODUCTION

We propose and analyze chaotic, asynchronous relaxations for the numerical solution by parallel processors of initial value problems involving a class of ordinary differential equations. The differential equations may be non-linear, non-autonomous and non-homogeneous. This paper gives analytical results on linear systems while results on nonlinear systems are reported in ref. 12. The proposed algorithm has several advantages in the parallel processing context, and these apply irrespective of the number of available processors and the architecture of the computing system. The main attractions are three-fold: first, it is essentially asynchronous in that it requires negligible synchronization or coordination between tasks, so that the idle times of processors are negligible; second, the chaotic computations stagger the data communication and memory accesses and reduce delays; finally, task management and programming are simplified. The algorithm also has advantages in the uniprocessor context.

Let $\|\mathbf{x}\|$ denote a norm of $\mathbf{x} \in \mathbb{R}^N$ and X the set of all N-vector functions which are continuous in the interval $[0, t']$. Define the norm of any function $\mathbf{x}$ in X to be $|\mathbf{x}| = \sup_{0 \leqslant t \leqslant t'} \|\mathbf{x}(t)\|$. We prove that $|\mathbf{x}^{(i)} - \mathbf{x}| \leqslant cR^i$, $0 \leqslant R < 1$, $c < \infty$, where $\mathbf{x}^{(i)}$ is the vector function available after the i^{th} update in the asynchronous relaxations, and $\mathbf{x}$ is the unique vector function solution of the given initial value problem. Thus uniform convergence at a geometric rate is proven. See Section 5.2 for a discussion and summary.

There are two sets of assumptions made to arrive at the above results. The first specifically concerns the asynchronous aspects and the second concerns the differential equations. Roughly, the first set (see Section 2.3) requires that the delays in the implementation are bounded by a finite constant and that every type of task is guaranteed at least one completion in all sequences of task completions of an arbitrary, but fixed, length. These requirements are generally easy to satisfy. The assumptions on the differential equations (see Sections 4.1) are all related to dominance conditions.

The iteration in the algorithm is on functions. From the point of view of this paper, the primitive tasks consist primarily of the computation of functions which are solutions of non-homogeneous, initial value problems. The method of numerical integration used to solve the problem is unrestricted and is not discussed in this paper. The work-load of a processor undertaking a primitive task consists of the following in the given order: reading the functions which are part of the definition of the task, solving the initial value problem associated with the task, and finally writing, or disseminating, the solution function, for instance, by writing into shared memory.

The asynchronous properties of the relaxations are derived primarily from two sources. First, the times taken to perform tasks are allowed to vary unpredictably within large bounds, and, second, great latitude is allowed in the sequencing of tasks. In particular, tasks do not require to be coordinated and, until termination, succeeding tasks are defined and ready to commence as soon as processors become available. In particular, therefore, the algorithm is not tied to the availability of any particular number of processors; indeed, the number is allowed to vary in the course of computations. Nor is the algorithm tied to any particular architecture of the computing system. The convergence result is for a computational model which accommodates major differences in memory, communications and synchronization primitives, [10], [11]. These differences are reflected in the values of parameters of the computational model, i.e. the delays and the compositions of the update sets.

The synchronous version of the algorithm, see Section 2.2, has sometimes been called "waveform relaxations" [6], and these have been successfully applied to the solution of VLSI circuit-equations by uniprocessors. The asynchronous relaxations are also of interest in uniprocessor implementations. In one particular scheme the processor considers each subsystem once in each cycle, and the relaxation of a subsystem utilizes only the most recently computed functions associated with the other subsystems. The corresponding relaxation on functions is like the Gauss-Seidel relaxations for linear equations. This relaxation scheme is subsumed in the model for asynchronous computations given in this paper.

2. MODEL OF ASYNCHRONOUS RELAXATIONS

2.1 A Class of Differential Equations

For the equations considered in this paper, the initial values are given and unique solutions are known to exist for the interval of interest, say $0 \leqslant t < t'$. The value of t' is ignored here since our analysis applies for all t'; however, it is recognized that t' may be an important parameter in the implementation since storage requirements alone may cause the interval of interest to be partitioned into several smaller intervals.

The equations are of the form

$$\frac{d}{dt}\mathbf{x}(t) + \mathbf{F}(\mathbf{x},t) = \mathbf{G}(\mathbf{x},t) + \mathbf{u}(t); \quad t \geqslant 0,\ \mathbf{x}(0) = \mathbf{x}_0, \tag{2.1}$$

in which $\mathbf{x}$, $\mathbf{u}$, $\mathbf{F}$, and $\mathbf{G}$ are elements of $\mathbb{R}^N$ for each value of t. In addition, it is assumed that $\mathbf{F}$ has a special structure:

$$\mathbf{F}(\mathbf{x},t) = \{\mathbf{F}_1(\mathbf{x}_1,t), \mathbf{F}_2(\mathbf{x}_2,t), \ldots, \mathbf{F}_n(\mathbf{x}_n,t)\}' \tag{2.2}$$

$\mathbf{G}$ may not have this special structure. Thus (2.1) allows the following partitioning into n subsystems which are indexed by j.

$$\left.\begin{aligned} &\frac{d}{dt}\mathbf{x}_j(t) + \mathbf{F}_j(\mathbf{x}_j,t) = \mathbf{G}_j(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_n,t) + \mathbf{u}_j(t), \quad t \geqslant 0;\ 1 \leqslant j \leqslant n, \\ &\mathbf{x}_j(0) = \mathbf{x}_{j,0} \end{aligned}\right\} \tag{2.3}$$

In (2.2) and (2.3); $\mathbf{x}_j$, $\mathbf{u}_j$, $\mathbf{F}_j$ and $\mathbf{G}_j$ are elements of $\mathbb{R}^{N_j}$ for each value of t, and $\sum_{j=1}^{n} N_j = N$.

The assumed structure is naturally associated with physical systems composed of subsystems, in which case the internal dynamical relations of the j^{th} subsystem are incorporated in the function $\mathbf{F}_j$, while the cross-couplings between variables of the j^{th} subsystem and the variables in the other subsystems are reflected in the function $\mathbf{G}_j$. The function $\mathbf{G}_j$ is allowed to have $\mathbf{x}_j$ as an argument because in certain non-linear physical systems a complete separation of variables may not be possible without defining very large subsystems.

In Section 4 the simpler linear counterpart of (2.3) is considered.

2.2 The Model for Asynchronous Computations

In order to describe the computational model we need to first introduce a set of primitive tasks, $\{T_j\}$, of various types. The subscript gives the type, of which there are as many as there are subsystems in the system of differential equations, namely, n. As this suggests, there is a close connection between tasks T_j and the j^{th} subsystem in (2.3). We have described these tasks as primitive because, from the perspective of this paper, we are not concerned with the manner of implementation of the tasks.

For a task T_j to be completely defined, its arguments $(\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_n)$ must be specified. These arguments are continuous vector functions of t, $t \geqslant 0$, and $\mathbf{z}_j \in \mathbb{R}^{N_j}$, $1 \leqslant j \leqslant n$, for each value of t. We say that $\mathbf{y} = T_j(\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_n)$ if $\mathbf{y}(t)$, $t \geqslant 0$, is the unique solution of the following initial value problem.

$$\left.\begin{aligned}&\frac{d}{dt}\mathbf{y} + \mathbf{F}_j(\mathbf{y},t) = \mathbf{G}_j(\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_n,t) + \mathbf{u}_j(t); \quad t \geqslant 0,\\ &\mathbf{y}(0) = \mathbf{x}_{j,0}\end{aligned}\right\} \qquad (2.4)$$

Thus apart from its arguments, the task T_j is tied to the j^{th} subsystem in (2.3). For this reason we sometimes refer to the completion of a task T_j as an update of the j^{th} subsystem.

In the following description of the model for asynchronous computations, the symbol i is reserved for indexing the updates, i.e. task completions. Typically, but not always, the completion of only one task comprises an update. To allow the more general case, $U(i)$, a subset of $\{1,2,\ldots,n\}$, denotes the types of tasks (or subsystem indices) whose completion comprise the i^{th} update. If the i^{th} update is due, perhaps in part only, to a task T_j being completed, i.e. $j \in U(i)$, then we denote the result of this completed task by $\mathbf{x}_j^{(i+1)}$.

The computational model is

$$\begin{aligned}\mathbf{x}_j^{(i+1)} &= T_j(\mathbf{x}_1^{(i-d(i,j,1))}, \mathbf{x}_2^{(i-d(i,j,2))}, \ldots, \mathbf{x}_n^{(i-d(i,j,n))}), && \text{if } j \in U(i)\\ &= \mathbf{x}_j^{(i)}, && \text{if } j \notin U(i), \end{aligned} \qquad (2.5)$$

for $i = 0,1,2,\cdots$.

The terms $\{d(i,j,k)\}$ are referred to as delays. The delays are not assumed to be known in advance, and in fact, they will have many components, such as numerical integration time, communication delays, memory access delays and software delays. The only restriction that will be placed on the delay values, see A2 in Section 2.3, is one of uniform boundedness.

Synchronous relaxations are obtained from (2.5) in the special case where $d(i,j,k) \equiv 0$ and $U(i) = \{1,2,\ldots,n\}$ for each i. The synchronous relaxations have also been called "waveform relaxations" in the literature [6] on VLSI circuit-equations solution techniques. Also, the relaxations in which $U(i) = \{(i \bmod n) + 1\}$, $i \geqslant 0$, and $d(i,j,k) \equiv 0$, may reasonably be called Gauss-Seidel; these relaxations may be implemented by only 1 processor.

To start the relaxations, we require the continuous vector functions $\mathbf{x}_j^{(0)}(t)$, $t \geqslant 0$, $1 \leqslant j \leqslant n$, which are arbitrary except that their initial values match the given initial values for the subsystems, i.e. $\mathbf{x}_j^{(0)}(0) = \mathbf{x}_{j,0}$. We assume that

$$\mathbf{x}_j^{(0)}(t) = \mathbf{x}_j^{(i)}(t),\ t \geqslant 0,\ 1 \leqslant j \leqslant n, \ \text{ for } i < 0\,. \qquad (2.6)$$

This convenient assumption places no additional restrictions on the implementation.

The tasks as defined here have not previously been considered in the literature on asynchronous algorithms. The model for asynchronous computations given in (2.5), augmented by appropriate definitions of the tasks, gives models closely related to those in refs. 1-5. Examples of mappings of realistic parallel computations to the model in (2.5) may be found in ref. 5.

2.3 Assumptions on the Asynchronous Relaxations

While additional assumptions will be made, in the course of the analysis, on the differential equations given in Section 2.1, there are only two assumptions on the model in (2.5) for asynchronous computations.

Assumption A1: There exists a finite d which uniformly bounds the delays, i.e.

$$0 \leqslant d(i,j,k) \leqslant d < \infty,\ i \geqslant 0,\ 1 \leqslant j \leqslant n,\ 1 \leqslant k \leqslant n\,. \qquad (2.7)$$

Assumption A2: There exists an integer s, $s < \infty$, such that every subsystem is updated at least once in every s consecutive updates.

Assumption A2 is a non-starvation condition. Since each update concerns at least one subsystem, violations of A2 can only occur if the ratio of updates for some pair of updates is allowed to become unbounded.

Different versions of A1 and A2 have been assumed in previous work on asynchronous numeric algorithms [1-5].

3. PRELIMINARIES

3.1 Differential Inequalities

The subsequent analysis makes considerable use of differential inequalities. For this reason we collect here certain well-known results related to differential inequalities; for additional information the reader is referred to the excellent text by Coppel [7].

Observe that where $x(t)$ is continuously differentiable, the derivative of $|x(t)|$ may not exist at points where $x(t) = 0$. However, the right derivative, $\frac{d^+}{dt}|x(t)|$, which is defined below, does exist.

$$\frac{d^+}{dt}|x(t)| \triangleq \lim_{h \to +0} \frac{|x(t+h)| - |x(t)|}{h} \tag{3.1}$$

To calculate $\frac{d^+}{dt}|x(t)|$ it is helpful to have the function $\operatorname{sgn}(x)$ where

$$\operatorname{sgn}(x) = 1 \quad \text{if } x > 0; \quad = -1 \quad \text{if } x < 0; \quad = 0 \quad \text{if } x = 0. \tag{3.2}$$

It then follows that

$$\frac{d^+}{dt}|x(t)| = \operatorname{sgn}(x(t))\,\frac{d}{dt}x \quad \text{if } x(t) \neq 0, \tag{3.3,i}$$

$$= \left|\frac{d}{dt}x(t)\right| \quad \text{if } x(t) = 0. \tag{3.3,ii}$$

The main result on differential inequalities that we will need is a result from Coppel [7]. This theorem applies to differential inequalities in which the right hand side is restricted to be of type K (after Kamke). A vector function $\mathbf{f} = (f_1, f_2, \cdots)$ of a vector variable $\mathbf{x} = (x_1, x_2, \cdots)$ is of type K in a set S if for any two vectors in S, $\mathbf{a}$ and $\mathbf{b}$,

$$\left.\begin{array}{l} a_p = b_p \\ a_q \leqslant b_q, \; q \neq p \end{array}\right\} \text{ implies } f_p(\mathbf{a}) \leqslant f_p(\mathbf{b}) .$$

For the example

$$\mathbf{f}(\mathbf{x},t) = \mathbf{M}(t)\mathbf{x} + \mathbf{v}(t) \tag{3.5}$$

where $\mathbf{M}$ is a continuous, matrix function of t and $\mathbf{v}(t)$ is a continuous vector function of t, $\mathbf{f}(\mathbf{x},t)$ is of type K for each fixed value of t if and only if all off-diagonal elements of $\mathbf{M}$ are non-negative. In the analysis to follow we shall be concerned with differential inequalities involving vector functions of the above kind.

Coppel [7] proves, in particular, that the following holds. Let $\mathbf{z}(t)$ be the unique solution on an interval $[a,b]$ of the equation

$$\frac{d}{dt}\mathbf{z}(t) = \mathbf{f}(\mathbf{z},t), \tag{3.6}$$

where $\mathbf{f}(\mathbf{z},t)$ is continuous in an open set, satisfies the Lipschitz condition [7], and is of type K for each fixed value of t. If $\mathbf{x}(t)$ is continuous on $[a,b]$, satisfies the differential inequality

$$\frac{d^+}{dt}\mathbf{x}(t) \leqslant \mathbf{f}(\mathbf{x},t) \tag{3.7}$$

on (a,b) and $\mathbf{x}(a) \leqslant \mathbf{z}(a)$, then $\mathbf{x}(t) \leqslant \mathbf{z}(t)$ for $a \leqslant t \leqslant b$.

3.2 Notation

All vectors are assumed to be column vectors. Vector transposition is denoted by the superscript $'$. For $\mathbf{x}$ a vector with a typical element denoted by x_p, let $|\mathbf{x}|$ denote the vector in which $|\mathbf{x}|_p = |x_p|$ for each p. In particular $\frac{d^+}{dt}|\mathbf{x}(t)| = \left\{\frac{d^+}{dt}|x_1(t)|, \frac{d^+}{dt}|x_2(t)|, \cdots\right\}'$. We employ the inequality convention between vectors in which $\mathbf{x} \leqslant \mathbf{y}$ is equivalent to $x_p \leqslant y_p$ for each p.

The above conventions are extended to matrices in the natural way. Thus $|\mathbf{M}|$ denotes the matrix obtained from the matrix $\mathbf{M}$ by replacing elements by their absolute values. Also, we let $\mathbf{I}$ denote the identity matrix and $\mathbf{e}_q$ the q^{th} column of $\mathbf{I}$. The vector $\mathbf{1}$ is defined to have unity for all its elements.

3.3 A Bound on the Fundamental Matrix of a System of Non-autonomous Linear Equations

Let $\mathbf{M}(t)$ be a continuous, matrix function of t, $t \geqslant 0$. Let $\mathbf{\Phi}_M(t)$ denote the corresponding fundamental matrix, i.e.,

$$\frac{d}{dt}\mathbf{\Phi}_M(t) = \mathbf{M}(t)\,\mathbf{\Phi}_M(t),\ t \geqslant 0;\ \mathbf{\Phi}_M(0) = \mathbf{I}\,. \tag{3.8}$$

Let $\mathbf{N}(t)$ be the matrix obtained from $\mathbf{M}(t)$ in the following manner,

$$\begin{aligned} N_{pq}(t) &= M_{pq}(t), \quad \text{if } p = q \\ &= |M_{pq}(t)|, \text{ if } p \neq q\,. \end{aligned} \tag{3.9}$$

Note that the vector function $\mathbf{f}(\mathbf{x},t) = \mathbf{N}(t)\mathbf{x}$ if of type K for every t. Finally, let $\mathbf{\Phi}_N(t)$ denote the fundamental matrix corresponding to $\mathbf{N}(t)$.

It can be shown [12] through a simple application of the aforementioned result on differential inequalities that the following holds.

Proposition 1:

$$|\mathbf{\Phi}_M(t)| \leqslant \mathbf{\Phi}_N(t),\ t \geqslant 0\,. \tag{3.10}$$

□

Corollary: For $\mathbf{N}(t)$ any continuous matrix function of t with non-negative off-diagonal elements for each value of t, the fundamental matrix corresponding to $\mathbf{N}(t)$, $\mathbf{\Phi}_N(t)$, is non-negative.

□

4. LINEAR DIFFERENTIAL EQUATIONS

The problem that we will consider here is the numerical computation of the unique solution $\mathbf{x}(t)$, $\mathbf{x}(t) \in \mathbb{R}^N$, $t \geqslant 0$, of a linear counterpart of the equations described in Section 2.1. If we let $\mathbf{F}_j(\mathbf{x}_j,t) = \mathbf{D}(j;t)\,\mathbf{x}_j$, and $\mathbf{G}_j(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, t) = \sum_{k=1}^{n} \mathbf{B}(j,k;t)\,\mathbf{x}_k(t)$, for $1 \leqslant j \leqslant n$, then we obtain from (2.3) the following system of partitioned differential equations.

$$\left.\begin{aligned} &\frac{d}{dt}\mathbf{x}_j(t) + \mathbf{D}(j;t)\,\mathbf{x}_j(t) = \sum_{k=1}^{n} \mathbf{B}(j,k;t)\,\mathbf{x}_k(t) + \mathbf{u}_j(t),\ t \geqslant 0,\ 1 \leqslant j \leqslant n\,. \\ &\mathbf{x}_j(0) = \mathbf{x}_{j,0}\,. \end{aligned}\right\} \tag{4.1}$$

The constituents of the given quantities, $\mathbf{D}$, $\mathbf{B}$, and $\mathbf{u}$, are continuous in $t, t \geqslant 0$, and bounded. The matrices $\mathbf{D}(j;t)$ and $\mathbf{B}(j,k;t)$ are respectively N_j by N_j and N_j by N_k.

We let $D_{pq}(j;t)$, $1 \leqslant p \leqslant N_j$, $1 \leqslant q \leqslant N_j$ denote the components of $\mathbf{D}(j;t)$. Similarly, $B_{pq}(j,k;t)$, $1 \leqslant p \leqslant N_j$, $1 \leqslant q \leqslant N_k$, denote the components $\mathbf{B}(j,k;t)$.

4.1 Assumption

We let $\mathbf{M}(j;t)$ denote the N_j by N_j matrix derived from $\mathbf{D}(j;t)$ in the following manner:

$$\begin{aligned} M_{pq}(j;t) &\triangleq D_{pq}(j;t), \qquad p = q, \\ &\triangleq -|D_{pq}(j;t)|, \quad p \neq q. \end{aligned} \tag{4.2}$$

The N by N matrix $\mathbf{M}(t)$ is then defined in the natural way:

$$\mathbf{M}(t) = \mathbf{M}(1;t) \oplus \mathbf{M}(2;t) \oplus \cdots \oplus \mathbf{M}(n;t) \tag{4.3}$$

Thus $\mathbf{M}(t)$ is block-diagonal with non-positive off-diagonal elements for each t.

The N_j by N_k matrix $\mathbf{N}(j,k;t)$ is derived from $\mathbf{B}(j,k;t)$: for $1 \leqslant j \leqslant n$, $1 \leqslant k \leqslant n$, $1 \leqslant p \leqslant N_j$, $1 \leqslant q \leqslant N_k$,

$$N_{pq}(j,k;t) = |B_{pq}(j,k;t)|\,. \tag{4.4}$$

Finally, the N by N matrix $\mathbf{N}(t)$ is defined to have $\mathbf{N}(j,k;t)$ as its $(j,k)^{\text{th}}$ block constituent.

In the analysis of linear differential equations we will need the following.

Assumption L1: There exists a constant N-vector $\boldsymbol{\pi}$ with all components finite and positive, and a positive constant ϵ such that

$$[\mathbf{M}(t) - \mathbf{N}(t)]\,\boldsymbol{\pi} \geqslant \epsilon \mathbf{1}, \quad t \geqslant 0\,. \tag{4.5}$$

Corollary 1: There exists a constant r, $0 \leqslant r < 1$, such that

$$\mathbf{N}(t)\boldsymbol{\pi} \leqslant r\mathbf{M}(t)\boldsymbol{\pi}, \quad t \geqslant 0\,. \tag{4.6}$$

Proof: From the prior assumption of absolute boundedness of the constituent functions of the matrices appearing in (4.1), it follows that there exists some b, $b < \infty$, such that

$$\mathbf{M}(t)\boldsymbol{\pi} \leqslant b\mathbf{1}, \quad t \geqslant 0\,. \tag{4.7}$$

Let $r = 1 - \epsilon/b$, so that $0 \leqslant r < 1$. With this identification it is straightforward to verify (4.6).

□

Corollary 2: Suppose that in (4.1), $\{\mathbf{D}(j;t)\}$ and $\{\mathbf{B}(j,k;t)\}$ are constant i.e. independent of t. Then Assumption L1 is equivalent to the condition

$$(\mathbf{M} - \mathbf{N}) \text{ is a } M\text{- matrix}\,. \tag{4.8}$$

Proof: $(\mathbf{M} - \mathbf{N})$ is a matrix with all off-diagonal elements non-positive. The assertion is a well-known property of such matrices [8]. (The reader will find in [9] a variety of other equivalent properties.)

□

When (4.1) is autonomous, it is possible to obtain a sharp estimate of the important constant r which appears in (4.6).

Corollary 3: If $\{\mathbf{D}(j;t)\}$ and $\{\mathbf{B}(j,k;t)\}$ in (4.1) are independent of t, then

(i) $\rho < 1$, where ρ is the spectral radius of $\mathbf{M}^{-1}\mathbf{N}$. (4.9,i)

(ii) For any $r > \rho$, there exists a positive vector $\boldsymbol{\pi}$ such that

$$\mathbf{N}\boldsymbol{\pi} < r\mathbf{M}\boldsymbol{\pi}\,. \tag{4.9,ii}$$

Proof: The proof of (i) is in [8]. The proof of (ii) follows from well-known properties of M-matrices: $\{r\mathbf{I} - \mathbf{M}^{-1}\mathbf{N}\}$ and $\mathbf{M}$ are M-matrices and consequently their inverses exist and are non-negative; hence the inverse of $(r\mathbf{M} - \mathbf{N})$ exists and is non-negative. Let $\boldsymbol{\pi} = (r\mathbf{M} - \mathbf{N})^{-1}\boldsymbol{\mu}$, where $\boldsymbol{\mu}$ is any positive vector. Clearly $\boldsymbol{\pi}$ is itself positive and also (4.9,ii) holds.

□

Note that in (ii) of the above Corollary, r may be taken to be arbitrarily close to ρ.

4.2 Asynchronous Relaxations

Recall that the symbol i is used to index updates, $U(i)$ denotes the set of indices of subsystems which are updated in the i^{th} update, and $\{\mathbf{x}_j^{(i+1)}(t),\ t \geqslant 0,\ 1 \leqslant j \leqslant n\}$ are the computed function values on completion of the i^{th} update. From (2.4), (2.5) and (4.1) we see that these functions are the solutions of the following equations

$$\left.\begin{aligned} &\frac{d}{dt}\mathbf{x}_j^{(i+1)}(t) + \mathbf{D}(j;t)\mathbf{x}_j^{(i+1)}(t) = \sum_{k=1}^{n}\mathbf{B}(j,k;t)\mathbf{x}_j^{(i-d(i,j,k))}(t) + \mathbf{u}_j(t)\,, \\ &t \geqslant 0;\ \mathbf{x}_j^{(i+1)}(0) = \mathbf{x}_{j,0}\,. \end{aligned}\right\} \text{ if } j \in U(i) \tag{4.10,i}$$

$$\mathbf{x}_j^{(i+1)}(t) \equiv \mathbf{x}_j^{(i)}(t),\ t \geqslant 0, \quad \text{if } j \notin U(i)\,. \tag{4.10,ii}$$

It is easy to see that a unique solution exists to the initial value problem in (4.10,i). The N_j-vector function $\mathbf{x}_j^{(i+1)}(t)$, $t \geqslant 0$, is the estimate of $\mathbf{x}_j(t)$, as given by (4.1), after the i^{th} update.

4.3 Analysis

4.3.1 Comparison System.

Define the N_j-vector $\mathbf{y}_j^{(i)}(t)$ thus,

$$\mathbf{y}_j^{(i)}(t) \triangleq \mathbf{x}_j^{(i)}(t) - \mathbf{x}_j(t), \quad t \geqslant 0, \quad 1 \leqslant j \leqslant n,\ i = 0, 1, \cdots \tag{4.11}$$

From (4.1) and (4.10) it is clear that

$$\mathbf{y}_j^{(i)}(0) = \mathbf{0}, \quad 1 \leqslant j \leqslant n, \quad i = 0,1, \cdots . \tag{4.12}$$

Also, from (4.1) and (4.10),

$$\frac{d}{dt}\mathbf{y}_j^{(i+1)}(t) + \mathbf{D}(j;t)\mathbf{y}_j^{(i+1)}(t) = \sum_{k=1}^{n} \mathbf{B}(j,k;t)\mathbf{y}_k^{(i-d(i,j,k))}(t), \quad \text{if } j \in U(i) \tag{4.13,i}$$

$$\mathbf{y}_j^{(i+1)}(t) \equiv \mathbf{y}_j^{(i)}(t), \quad t \geqslant 0, \quad \text{if } j \notin U(i)\,. \tag{4.13,ii}$$

4.3.2 Differential Inequalities. Let $y_{j,p}^{(i+1)}(t)$ denote the p^{th} constituent of the N_j-vector $\mathbf{y}_j^{(i+1)}(t)$. From (4.13,i), for $j \in U(i)$,

$$\left\{\operatorname{sgn} y_{j,p}^{(i+1)}(t)\right\}\frac{d}{dt}y_{j,p}^{(i+1)}(t) + D_{pp}(j;t)\left|y_{j,p}^{(i+1)}(t)\right| + \sum_{q:q\neq p} D_{pq}(j;t)\left[\left\{\operatorname{sgn} y_{j,p}^{(i+1)}(t)\right\} y_{j,q}^{(i+1)}(t)\right]$$

$$= \sum_{k=1}^{n}\sum_{q=1}^{N_q} B_{pq}(j,k;t)\left[\left\{\operatorname{sgn} y_{j,p}^{(i+1)}(t)\right\} y_{k,q}^{(i-d(i,j,k))}(t)\right], \quad t \geqslant 0\,. \tag{4.14}$$

It follows that

$$\left\{\operatorname{sgn} y_{j,p}^{(i+1)}(t)\right\}\frac{d}{dt}y_{j,p}^{(i+1)}(t) + D_{pp}(j;t)\left|y_{j,p}^{(i+1)}(t)\right| - \sum_{q:q\neq p}\left|D_{pq}(j;t)\right|\left|y_{j,q}^{(i+1)}(t)\right|$$

$$\leqslant \sum_{k=1}^{n}\sum_{q=1}^{N_q}\left|B_{pq}(j,k;t)\right|\left|y_{k,q}^{(i-d(i,j,k))}(t)\right|, \quad t \geqslant 0\,. \tag{4.15,i}$$

Also, when $y_{j,p}^{(i+1)}(t) = 0$ it follows directly from (4.13,i) that

$$\left|\frac{d}{dt}y_{j,p}^{(i+1)}(t)\right| - \sum_{q:q\neq p}\left|D_{pq}(j;t)\right|\left|y_{j,q}^{(i+1)}(t)\right| \leqslant \sum_{k=1}^{n}\sum_{q=1}^{N_q}\left|B_{pq}(j,k;t)\right|\left|y_{k,q}^{(i-d(i,j,k))}(t)\right| \tag{4.15,ii}$$

Hence, from the definition of the right derivative, see (3.3) in Sec. 3.1, it follows from (4.2), (4.3) and (4.4) that

$$\frac{d^+}{dt}\left|\mathbf{y}_j^{(i+1)}(t)\right| + \mathbf{M}(j;t)\left|\mathbf{y}_j^{(i+1)}(t)\right| \leqslant \sum_{k=1}^{n}\mathbf{N}(j,k;t)\left|\mathbf{y}_k^{(i-d(i,j,k))}(t)\right|, \tag{4.16}$$

$$t \geqslant 0\,.$$

In summary, if $j \in U(i)$ then (4.16) augmented by the initial condition $\left|\mathbf{y}_j^{(i+1)}(0)\right| = \mathbf{0}$ holds, and if $j \notin U(i)$ then

$$\left|\mathbf{y}_j^{(i+1)}(t)\right| = \left|\mathbf{y}_j^{(i)}(t)\right|, \quad t \geqslant 0\,. \tag{4.17}$$

4.3.3 Bounds. Making use of (4.6) in Corollary 1 to Assumption L1, the right hand side of (4.16) may be upper bounded thus: for $t \geqslant 0$,

$$\sum_{k=1}^{n}\mathbf{N}(j,k;t)\left|\mathbf{y}_k^{(i-d(i,j,k))}(t)\right| \leqslant \left[r\left\{\max_{1\leqslant k\leqslant n} w_k^{(i-d(i,j,k))}\right\}\right]\mathbf{M}(j;t)\,\boldsymbol{\pi}_j \tag{4.18}$$

where

$$w_j^{(i)} \triangleq \sup_{t\in[0,\infty]}\ \max_{1\leqslant p\leqslant N_j}\left\{\left|y_{j,p}^{(i)}(t)\right|/\pi_{j,p}\right\}, \quad 1 \leqslant j \leqslant n,\ i \geqslant 0\,. \tag{4.19}$$

From (4.16) and (4.18) we then have, for $j \in U(i)$, $t \geqslant 0$,

$$\left.\begin{aligned}
&\frac{d^+}{dt}\left\{\left|\mathbf{y}_j^{(i+1)}(t)\right| - s_j^{(i+1)}\,\boldsymbol{\pi}_j\right\} + \mathbf{M}(j;t)\left\{\left|\mathbf{y}_j^{(i+1)}(t)\right| - s_j^{(i+1)}\,\boldsymbol{\pi}_j\right\} \leqslant 0, \qquad t \geqslant 0\,.\\
&\left\{\left|\mathbf{y}_j^{(i+1)}(0)\right| - s_j^{(i+1)}\,\boldsymbol{\pi}_j\right\} = -s_j^{(i+1)}\,\boldsymbol{\pi}_j
\end{aligned}\right\} \tag{4.20}$$

where,

$$s_j^{(i+1)} \triangleq r\left\{ \max_{1 \leqslant k \leqslant n} w_k^{(i-d(i,j,k))} \right\}. \tag{4.21}$$

Now, if the differential inequality in (4.20) is stated in the form

$$\frac{d^+}{dt}\, \mathbf{z}(t) \leqslant \mathbf{f}(t,\mathbf{z}), \quad t \geqslant 0 \tag{4.22}$$

then **f** is of type K for each t, see Sec. 3.1. The prime reason for this is that the off-diagonal terms of $\mathbf{M}(j;t)$ are non-positive. An application of the main result on differential inequalities stated in Sec. 3.1 then yields

$$\left| \mathbf{y}_j^{(i+1)}(t) \right| - s_j^{(i+1)} \boldsymbol{\pi}_j \leqslant \mathbf{z}(t), \; t \geqslant 0 \tag{4.23}$$

where $\mathbf{z}(t)$ is the unique solution of the following initial value problem:

$$\left.\begin{aligned} &\frac{d}{dt}\mathbf{z}(t) + \mathbf{M}(j;t)\, \mathbf{z}(t) = 0, \; t \geqslant 0 \\ &\mathbf{z}(0) = -s_j^{(i+1)} \boldsymbol{\pi}_j \, . \end{aligned}\right\} \tag{4.24}$$

But the solution of (4.24) is

$$\mathbf{z}(t) = \boldsymbol{\Phi}(t)\, \mathbf{z}(0), \; t \geqslant 0 \tag{4.25}$$

where $\boldsymbol{\Phi}(t)$ is the fundamental matrix corresponding to $-\mathbf{M}(j;t)$. Furthermore, since the off-diagonal elements of the latter matrix are non-negative, we may conclude from the Corollary to Proposition 1 in Sec. 3.3 that $\Phi(t)$ is non-negative. Also, $\mathbf{z}(0)$ as specified in (4.24) is non-positive. It therefore follows that

$$\mathbf{z}(t) \leqslant \mathbf{0}, \; t \geqslant 0 \, . \tag{4.26}$$

Hence, from (4.23) and (4.26),

$$\left| \mathbf{y}_j^{(i+1)}(t) \right| \leqslant s_j^{(i+1)} \boldsymbol{\pi}_j, \; t \geqslant 0 \, .$$

From the definitions of $w_j^{(i+1)}$ and $s_j^{(i+1)}$ in (4.19) and (4.21), we obtain the result which is summarized now.

Proposition 2: For the linear differential equations in (4.1) subject to Assumption L1, the asynchronous relaxations in (4.10) satisfy the following: for $i = 0,1,2, \cdots$

$$w_j^{(i+1)} \leqslant r\left\{ \max_{1 \leqslant k \leqslant n} w_k^{(i-d(i,j,k))} \right\} \quad \text{if } j \in U(i), \tag{4.27,i}$$

$$= w_j^{(i)} \quad \text{if } j \notin U(i) \tag{4.27,ii}$$

where, see (4.6), $0 \leqslant r < 1$, and

$$w_j^{(i)} = \sup_{t \in [0,\infty]} \max_{1 \leqslant p \leqslant N_j} \left\{ \left| y_{j,p}^{(i)}(t) \right| / \pi_{j,p} \right\},$$

and

$$y_{j,p}^{(i)}(t) = x_{j,p}^{(i)}(t) - x_{j,p}(t) \, .$$

□

5. CONVERGENCE OF THE ASYNCHRONOUS RELAXATIONS

The above result relies only on assumptions on the respective differential equations and not at all on those aspects of the asynchronous process which have to do with the relative frequencies of updates and the delays in the updates. Here we introduce these aspects and, by making use of Assumptions A1 and A2 given in Sec. 2, give a proof of uniform convergence.

5.1 Convergence of the Discrete, Asynchronous Process

From Proposition 2,

$$w_j^{(i+1)} \leqslant r\left\{\max_{1\leqslant k\leqslant n} w_k^{(i-d(i,j,k))}\right\} \quad \text{if } j \in U(i)$$

$$= w_j^{(i)} \qquad \text{if } j \notin U(i)\,.$$

Hence from Assumption A1,

$$w_j^{(i+1)} \leqslant r\left[\max_{0\leqslant\delta\leqslant d}\ \max_{1\leqslant k\leqslant n} w_k^{(i-\delta)}\right] \quad \text{if } j \in U(i). \tag{5.1}$$

By recalling that $r < 1$ we have, in particular,

$$\max_{0\leqslant\delta\leqslant d}\ \max_{1\leqslant j\leqslant n} w_j^{(i'-\delta)} \leqslant \max_{0\leqslant\delta\leqslant d}\ \max_{1\leqslant j\leqslant n} w_j^{(i-\delta)}, \quad 0 \leqslant i \leqslant i'-1\,. \tag{5.2}$$

We also have [12]

Proposition 3:

$$\max_{1\leqslant j\leqslant n} w_j^{(i')} \leqslant r\left[\max_{0\leqslant\delta\leqslant d}\ \max_{1\leqslant j\leqslant n} w_j^{(i-\delta)}\right], \quad 0 \leqslant i \leqslant i'-s\,. \tag{5.3}$$

□

The above Proposition has the following Corollary in the form of a recursive inequality:

$$\left[\max_{0\leqslant\delta\leqslant d}\ \max_{1\leqslant j\leqslant n} w_j^{(i'-\delta)}\right] \leqslant r\left[\max_{0\leqslant\delta\leqslant d}\ \max_{1\leqslant j\leqslant n} w_j^{(i-\delta)}\right], \quad 0 \leqslant i \leqslant i'-(d+s) \tag{5.4}$$

We now state the main result, in which the norm $\|\ \|$ in $\mathbb{R}^N$ is,

$$\|\mathbf{x}\| = \max_{1\leqslant j\leqslant n}\ \max_{1\leqslant p\leqslant N_j} |x_{j,p}|/\pi_{j,p}\,.$$

Proposition 4: For $i = 0,1,2,\cdots$

$$\sup_{t\in[0,\infty]} \|\mathbf{x}^{(i)}(t) - \mathbf{x}(t)\| \leqslant Cr^{\ell}, \tag{5.5}$$

where $0 \leqslant r < 1$, $\ell = \lfloor i/(d+s) \rfloor$ and $C = \max\limits_{1\leqslant j\leqslant n} w_j^{(0)}$.

Proof: (5.4) yields

$$\left[\max_{0\leqslant\delta\leqslant d}\ \max_{1\leqslant j\leqslant n} w_j^{(i-\delta)}\right] \leqslant Cr^{\ell} \tag{5.6}$$

where,

$$C = \max_{0\leqslant\delta\leqslant d}\ \max_{1\leqslant j\leqslant n} w_j^{(-\delta)} = \max_{1\leqslant j\leqslant n} w_j^{(0)}\,.$$

The statement in the Proposition is implied by (5.6).

□

5.2 Discussion

We may conclude from the last Proposition that the long-term average rate of geometric convergence (per update) is not less than $r^{1/(d+s)}$.

We have proven uniform convergence for the proposed asynchronous relaxations under different conditions for the following variants of the differential equations in (2.1):

$$(i)\ \frac{d}{dt}\mathbf{x}(t) + \mathbf{D}\mathbf{x}(t) = \mathbf{B}\mathbf{x}(t) + \mathbf{u}(t)\,. \tag{5.7,i}$$

$$(ii)\ \frac{d}{dt}\mathbf{x}(t) + \mathbf{D}(t)\mathbf{x}(t) = \mathbf{B}(t)\mathbf{x}(t) + \mathbf{u}(t)\,. \tag{5.7,ii}$$

We have assumed A1 and A2, see Section 2.3, in all cases. In case (i) uniform convergence at a geometric rate occurs when $\mathbf{M}^{-1}\mathbf{N}$ is a M-matrix; in this case the parameter r in the bound on the rate of convergence may be chosen to be arbitrarily close to ρ, the spectral radius or $\mathbf{M}^{-1}\mathbf{N}$, where $\mathbf{M}$ and $\mathbf{N}$ are obtained from $\mathbf{D}$ and $\mathbf{B}$ as in (4.2) and (4.4). In particular, if $\mathbf{D}$ has non-positive off-diagonal elements and $\mathbf{B}$ is non-negative, then $\mathbf{M} = \mathbf{D}$ and $\mathbf{N} = \mathbf{B}$.

The uniform convergence for case (ii) requires Assumption L1, see Section 4.1.

REFERENCES

[1] D. Chazan and W. Miranker, "Chaotic Relaxation," Linear Algebra and Its Applications, **2**, 1969, pp. 199-222.

[2] G. M. Baudet, "Asynchronous Iterative Methods for Multiprocessors," J. of the ACM, **25**, No. 2, April 1978, pp. 226-244.

[3] D. P. Bertsekas, "Distributed Asynchronous Computation of Fixed Points," Math. Programming, **27**, 1983, pp. 107-120.

[4] D. P. Bertsekas, "Distributed Dynamic Programming," IEEE Trans. Automatic Control, **AC-27**, No. 3, June 1982, pp. 610-616.

[5] B. Lubachevsky and D. Mitra, "A Chaotic, Asynchronous Algorithm for Computing the Fixed Point of a Nonnegative Matrix of Unit Spectral Radius," J. of the ACM, **33**, No. 1, January 1986.

[6] E. Lelarasmee, A. E. Ruehli and A. L. Sangiovanni-Vincentelli, "The Waveform Relaxation Method for Time-Domain Analysis of Large Scale Integrated Circuits," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, **CAD-1**, No. 3, July 1982 pp. 131-145.

[7] W. A. Coppel, "Stability and Asymptotic Behavior of Differential Equations," Heath Mathematical Monographs, Boston, 1965.

[8] R. S. Varga, "Matrix Iterative Analysis," Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

[9] M. Fiedler and V. Ptak, "On Matrices with Nonpositive Off-diagonal Elements and Positive Principal Minors," Czech. Math. J., **12**, No. 3, 1962, pp. 382-400.

[10] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph and M. Snir, "The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer", IEEE Trans. Computers, Vol. C-32, No. 2, Feb. 1983, pp. 175-189.

[11] L. A. Cohn, "A Conceptual Approach to General Purpose Parallel Computer Architecture", Thesis, Columbia University, 1983.

[12] D. Mitra, "Chaotic, Asynchronous Relaxations for the Numerical Solution of Differential Equations by Parallel Processors," AT&T Bell Laboratories Tech. Memo., Oct. 1984.

SCHEDULING DATA TRANSFERS IN NETWORKS WITH PRE-EMPTION

HYEONG-AH CHOI and S. LOUIS HAKIMI

Department of Electrical Engineering and Computer Science
Northwestern University
Evanston, IL 60201

ABSTRACT

The scheduling of data transfers in networks, where the schedule does not permit interruption and each communication module can be used as a transmitter and as a receiver, was studied by Coffman, et al. The same problem when interruption in the schedule is permitted and the transmitting and receiving modules are distinct was studied by Choi and Hakimi among others. Hajek studied another interesting variation of the problem where interruption is permitted but each communication module is a transmitter and a receiver. This paper presents certain generalizations and improvements of Hajek's results.

1. INTRODUCTION

In this paper we study a problem of transferring data between various pairs of nodes in a network. The nodes of this network represent communication centers (computers) and each node is capable of communicating directly with every other node. Let $G(V,E)$ be a weighted graph whose vertices, V, correspond to the nodes in the network, and where an edge $e=(u,v)$ is in E if and only if there is data to be transferred between nodes u and v. The weight of edge e, denoted by $w(e)$, represents the amount of data to be transferred between nodes u and v. To each vertex v in V there is assigned a positive integer, $\alpha(v)$, representing the number of communication modules at node v in the network. It is noted that each communication module can be used as a transmitter and as a receiver, which means that a module at u and a module at v must be employed to transfer data between nodes u and v. Our problem is to find a schedule of the data transfer engagements and the durations of these engagements so as to minimize the overall finishing time. Our graph terminology will be standard and for the undefined terms the reader is referred to [1].

We call the edge and vertex weighted graph $G(V,E)$ a "communication graph". Consider an integer $k \geq 2$ and an edge (u,w) in E. Assume, $k \leq \alpha(u)$ and $k \leq \alpha(w)$. If k communication modules at u and at w are allowed to simultaneously engage in

---

This work is supported by the National Science Foundation, Division of Electric, Computer and Systems Engineering, Computer Engineering Program, Grant ECS-84-06854.

transferring data between u and w, then we say "speed-up" is permitted. On the other hand, if at each instant of time at most two modules (one receiving and one transmitting) can be involved in performing a data transfer, then we say speed-up is not permitted. It is assumed that if speed-up is used using k communication modules at u and w, it would increase the speed of the transfer of data between u and w by a factor of k. Without loss of generality, for $e = (u, w)$ in G, $w(e)$ is assumed to represent the time it takes to transfer the entire data between u and w if no speed-up is used. We also assume that "pre-emption" is permitted (i.e., the task of transferring data between two nodes may be interrupted).

Various special cases of this problem have been studied previously [2,3,4,5]. Coffman, Garey, Johnson, and La Paugh [2] studied a version of this problem when pre-emption is not allowed. Another case of the problem, where the transmitting and receiving modules are distinct and pre-emption is permitted was studied by Choi and Hakimi [3] and by Gopal, Bongiovanni, Bonuccelli, Tang, and Wong [4]. Hajek [5] studied a new version of the problem, where pre-emption is allowed and each communication module is used as a transmitting and as a receiving module while assuming that each node in the network has one communication module. It should be noted that Hajek's version of the problem is reduced to the well-known edge-coloring problem [6], if the amount of data to be transferred between any pair of nodes is an integer and the durations of the engagements of the data transfers are all one. If we allow more than one communication module at each node of the network with the remaining assumption being as in the previous sentence, then the Hajek's version of the problem can be analyzed by a generalization of the edge-coloring problem studied by Hakimi and Kariv [7].

Hajek [5] showed that Edmonds' theorem [8] can be modified to prove that one can find a lower bound T on the finishing time of the schedule and show that there exists a schedule with finishing time T, when $\alpha(v) = 1$ for all $v \in V$. In the next section, we will show that Edmonds' theorem can also be used to generalize the result of Hajek to the case when $\alpha(v)$ is any positive integer for $v \in V$. Hajek's as well as our results are of existance type, that is we do not have, in general, a polynomial time algorithm for finding such a schedule.

In Section 3, we provide approximate algorithms of complexity

$$0\Big(|E|(|E| + \sum_{v\in V} \alpha(v)) \sum_{v\in V} \alpha(v)\Big) \text{and } 0(|E|^2 \sum_{v\in V} \alpha(v))$$

which produce schedules whose overall finishing times are no larger than $\ell_s/(\ell_s - 1)$ times the optimum finishing time, where ℓ_s is the length of the shortest odd cycle in $G(V, E)$, for the cases when speed-up is permitted or not.

Finally, it is simple to see that if $G(V,E)$ is bipartite, then the problem is reduced to the case studied by Choi and Hakimi [3] for which an optimal schedule can be found in polynomial time. However, we will show that if $\alpha(v) =$ even for all $v \in V$, then our problem is also reduced to the version of the problem which was studied in [3] when speed-up is permitted.

2. OPTIMUM FINISHING TIME

Let $G(V,E)$ be a weighted graph with $\alpha(v)$ being the number of communication modules at $v \in V$ and $w(e) = w(u,v)$ (with $(u,v) = e$) being the amount of time required to transfer the data between u and v. For $v \in V$, let $w(v)$ represent the sum of the weights of the edges incident at v. For any subgraph H of G, let $w\big(E(H)\big) = \sum\limits_{e \in E(H)} w(e)$.

The following theorem was first proved by Hajek [5] using a result of Edmonds [8].

Theorem 1. Let $G(V,E)$ be a communication graph with $\alpha(v) = 1$ for all $v \in V$. Then there exists a schedule for G, whose overall finishing time is

$$\max\left\{ \max\{w(v) | v \in V\},\ \max\left\{ \frac{w(E(H))}{\lfloor |V(H)|/2 \rfloor} \;\Big|\; H \subseteq G \right\} \right\} \left(\overset{\Delta}{=} T_0\right).$$

There exists no schedule whose overall finishing time is $< T_0$.

For any subgraph H of G, let $w_{\max}(H) = \max\{w_H(v) | v \in V(H)\}$, where $w_H(v)$ represents the sum of the weights of the edges of H incident at v. For any two subgraphs H_1 and H_2 of G, let $E\big(V(H_1), V(H_2)\big) = \big\{(u,v) | u \in V(H_1) \text{ and } v \in V(H_2)\big\}$ and let $w\Big(E\big(V(H_1), V(H_2)\big)\Big) = \sum \big\{w(e) | e \in E\big(V(H_1), V(H_2)\big)\big\}$.

Theorem 2. Let $G(V,E)$ be a communication graph. Then there exists a schedule for G with speed-up whose overall finishing time is

$$\max\left\{ \max\left\{ \frac{w(v)}{\alpha(v)} \;\Big|\; v \in V \right\},\ \max\left\{ \frac{w(E(H))}{\left\lfloor \sum\limits_{v \in V(H)} \alpha(v) \,/\, 2 \right\rfloor} \;\Big|\; H \subseteq G \right\} \right\} \left(\overset{\Delta}{=} T_0\right).$$

There is no such a schedule whose overall finishing time is $< T_0$.

The proof of Theorem 2 will be found in the full paper.

Let $d_H(v)$ be the degree of vertex $v \in V(G)$ in the subgraph H. Let $\alpha'_H(v) = \min\{d_H(v), \alpha(v)\}$ for $v \in V(G)$ and $H \subseteq G$.

Theorem 3: Let $G(V,E)$ be a communication graph. Then there exists a schedule for G without speed-up whose overall finishing time is

$$\max\left\{ \max_{e\in E(G)}\{w(e)\},\ \max_{v\in V(G)},\left\{\frac{w(v)}{\alpha(v)}\right\},\max_{H\subseteq G}\left\{\frac{w\big(E(H)\big)}{\left\lfloor \sum\limits_{v\in V(H)} \alpha'_H(v)\ /\ 2\right\rfloor}\right\}\right\}\left(\overset{\Delta}{=} T_0\right).$$

There is no such a schedule whose overall finishing time is $< T_0$.

The proof of Theorem 3 will be found in the full paper.

3. APPROXIMATION ALGORITHMS

3.1 Basic Lemma

Let $G(V, E)$ be a communication graph and let $w(e), w(v), w\big(E(H)\big)$ be as defined earlier. Let $w\prime(v) = w(v)/\alpha(v)$, $w'_m = \max\{w'(v)|v \in V\}$, and $V_m = \{v \in V|w'(v) = w'_m\}$. Let $d_G(v)$ be the degree of vertex $v \in V$ in graph G. Let $H(V, E_1)$ be an unweighted subgraph of G with edge set $E_1 \subseteq E$. We label each edge e in H with a positive number $\beta_H(e) \in \{\frac{1}{2}, 1\}$. A β-labeling is a proper labeling if the set of $E(\frac{1}{2}) = \{e \in E_1|\beta_H(e) = 1/2\}$ induces a subgraph of G consisting of vertex-disjoint union of odd cycles and $\beta_H(v) \leq \alpha(v)$ for all $v \in V$, where $\beta_H(v) = \sum\{\beta_H(e)|e \in E_1$ is incident at $v\}$. (In general, it may be difficult to determine whether a β-labeling is a proper labeling or not. But in the following lemma as well as the algorithms which follow it, the proper labeling of subgraph H is obtained while H itself is being constructed. In our case each edge of label $1/2$ belongs to exactly one odd cycle.)

The main result of this subsection is the following lemma.

Lemma 1: Let $G(V, E)$ be a weighted graph which may have parallel (multiple) edges. Assume that $0 < w(e) < w'_m$ for all $e \in E$ and $\alpha(v) \leq d_G(v)$ for all $v \in V$. Then there exists a subgraph H of G with proper β-labeling of H such that $\beta_H(v) \leq \alpha(v)$ for all $v \in V$, $\beta_H(v) = \alpha(v)$ for all $v \in V_m$, and if $\beta_H(e) = 1/2$ for $e = (u, w)$, then $u, w \in V_m$.

Remarks: Observe that if $\alpha(v) = 1$ for all $v \in V$, then the above lemma is reduced to Hajek's lemma [5].

The proof of Lemma 1 will be found in the full paper.

3.2 An Approximation Algorithm without Speed-up

We begin with the statement of the algorithm.

Algorithm 1

Step 0.

We have graph $G(V, E)$, positive integers $\alpha(v)$ for all $v \in V$ and positive numbers $w(e)$ for all $e \in E$. Set $i \leftarrow 0$, $t \leftarrow 0$, $w_{\max}(e) \leftarrow \max\{w(e)|e \in E\}$, and $w'_m \leftarrow$

$\max\{w(v)/\alpha(v) | v \in V\}$. Compute $d_G(v)$ for all $v \in V$. (We assume that $d_G(v) \geq \alpha(v)$ for all $v \in V$.)

Step 1.

(a) Set $i \leftarrow i + 1$, $s_i \leftarrow 0$, and $E'_i \leftarrow \emptyset$. For each $v \in V$, if $d_G(v) < \alpha(v)$, set $\alpha(v) \leftarrow d_G(v)$.

(b) Compute, for each $v \in V$, $w(v)$ (= the sum of the weights of the edges of G incident at v), $w'(v) = w(v)/\alpha(v)$, and $w'_{m_i} = \max\{w'(v) | v \in V\}$.

(c) If $w(e) < w'_{m_i}$ for all $e \in E$ GO TO Step 2(a).

(d) For each edge $e = (u, w)$ with $w(e) \geq w'_{m_i}$, set $E \leftarrow E - \{e\}$, $E'_i \leftarrow E'_i \cup \{e\}$, $\alpha(u) \leftarrow \alpha(u) - 1$, $\alpha(w) \leftarrow \alpha(w) - 1$, $d_G(u) \leftarrow d_G(u) - 1$, and $d_G(w) \leftarrow d_G(w) - 1$. If $d_G(v) = 0$, set $V \leftarrow V - \{v\}$. Repeat Step 1(d) until no such an edge exists. If $E = \emptyset$, GO TO Step 3(a), otherwise return to Step 1(b). (The deleted edges E'_i correspond to the data transfers that become engaged at time t and will remain engaged until the schedule is completed. The communication modules engaged to do these transfers will not be assigned any other tasks in the future.)

Step 2.

(a) $V_{1i} = \{v \in V | w'(v) = w'_{m_i}\}$.

(b) Compute subgraph H_i of G with proper labeling of H_i such that $\beta_{H_i}(v) \leq \alpha(v)$ for all $v \in V$ and $\beta_{H_i}(v) = \alpha(v)$ for all $v \in V_{1i}$. Compute $s_i = \min\Big\{\min\{\frac{w(e)}{\beta_{H_i}(e)} | e \in E(H_i)\}, \min\{w'_{m_i} - w(e) | e \in E - E(H_i)\}, \min\{2(w'_{m_i} - w(e)) | e \in E(H_i) \text{ and } \beta_{H_i}(e) = \frac{1}{2}\}, \min\{\frac{\alpha(v) \cdot w'_{m_i} - w(v)}{\alpha(v) - \beta_{H_i}(v)} \mid v \in V - V_{1i}\}\Big\}$. For each $e \in E(H_i)$ with $e = (u, w)$, set $w(e) \leftarrow w(e) - s_i \cdot \beta_{H_i}(e)$ and if $w(e) = 0$, set $e \leftarrow E - \{e\}$, $d_G(u) \leftarrow d_G(u) - 1$, and $d_G(w) \leftarrow d_G(w) - 1$. If $d_G(v) = 0$, set $V \leftarrow V - \{v\}$.

Step 3.

(a) For each edge $e \in E'_i$, start the data transfer corresponding to the edge e at time t until that transfer is completed.

(b) For each edge $e \in E(H_i)$ with $\beta_{H_i}(e) = 1$, start the data transfer corresponding to the edge e at time t for s_i seconds.

(c) A schedule for each edge $e \in E(H_i)$ with $\beta_{H_i}(e) = \frac{1}{2}$ is defined as follows.

Let $e = (v_p, v_{p+1})$ belong to an odd cycle denoted by a set of edges (v_0, v_1), (v_1, v_2), …, $(v_{2\ell}, v_0)$. Then the data transfer corresponding to the edge e become employed at time $t + p \cdot (s_i/2\ell)$ for $s_i/2\ell$ seconds and at time $t + \Big((p - (-1)^q \cdot q) \bmod (2\ell + 1)\Big) s_i/2\ell$ for $s_i/2\ell$ seconds, for $q = 2, 3, \ldots, \ell$.

(d) Compute $t_i = s_i * \frac{|c_{i_s}|}{|c_{i_s}|-1}$, where c_{i_s} represents the shortest odd cycle in H_i with $\beta_{H_i}(e_s) = \frac{1}{2}$ for all $e_s \in E(c_{i_s})$.

Set $t \leftarrow t + t_i$.

If $w'_{m_i} - s_i > 0$, GO TO Step 1(a).

Step 4.

STOP.

Theorem 4. Let $G(V,E)$, be a communication graph and let $d_G(v)$, $w(e)$, $w(v)$, $w'(v)$ and w'_m be as defined earlier. Assume that $d_G(v) \geq \alpha(v)$ for all $v \in V$. Then we have:

(a) the algorithm produces a correct schedule with finishing time (timespan)

$$t^* \leq \max\{\max\{w(e)|e \in E\},\ \ell_s/(\ell_s - 1) \cdot w'_m\},$$

where ℓ_s represents the length of the shortest odd length cycle in G and this does not exceed $\ell_s/(\ell_s - 1)$ times the optimum finishing time and

(b) the time complexity of our algorithm is

$$0(|E|^2 \sum_{v \in V} \alpha(v)).$$

The proof of Theorem 4 will be found in the full paper.

3.3 An Approximation Algorithm with Speed-up

In this section, we will consider the problem when speed-up is permitted. Let $G(V,E)$ be a weighted graph which represents our communication graph. Let t be a real positive variable. We construct a multigraph $G_t(V, E_t)$ as follows. For each $e \in E$, with say $e = (u, w)$, there are $\lceil \frac{w(e)}{t} \rceil$ edges in G_t which join the vertices u and w. All edges in G_t are assumed to have weight t. Now it is easy to see that one can use edge-coloring of G_t to find an approximate solution to our problem in this case, particularly when $\alpha(v) = 1$ for all $v \in V$. This is because the data transfers corresponding to the edges having say color i can be assigned the time interval $[(i-1)t, it]$. Nishizeki and Kashiwagi [9] obtained a new upper bound for the chromatic index of multigraph G:

$$\chi'(G) \leq \max\Big\{ \lfloor \frac{1}{10}(11\Delta(G) + 8) \rfloor,\ \max_{H \subseteq G} \Big\{ \Big\lceil \frac{|E(H)|}{\lfloor |V(H)|/2 \rfloor} \Big\rceil \Big\} \Big\},$$

where $\Delta(G) = \max\{d(v)|v \in V(G)\}$ and $E(H)$ (respectively, $V(H)$) represents the set of edges (respectively, vertices) of subgraph H of G. From the above result we obtain

$$\chi'(G_t) \le \max\left\{\frac{1}{10}\left(11\Delta(G) + \frac{11w_m}{t} + 8\right),\ \max_{H\subseteq G}\left\{\frac{|E(H)| + w(E(H))/t}{\lfloor V(H)/2\rfloor}\right\}\right\},$$

where $\Delta(G)$, $E(H)$, and $V(H)$ are as defined above, $w_m = \max\{w(v) | v \in V(G)\}$ and $w(E(H)) = \sum\{w(e) | e \in E(H)\}$.

Then the overall finishing time for G is

$$\chi'(G_t)\cdot t \le \max\left\{\frac{1}{10}(11\Delta(G)\cdot t + 11w_m + 8t),\ \max_{H\subseteq G}\left\{\frac{|E(H)|\cdot t + w(E(H))}{\lfloor V(H)/2\rfloor}\right\}\right\}.$$

We note that if t becomes very small, then the overall finishing time for G is

$$\chi'(G_t)\cdot t \le \max\left\{\left(\frac{11\cdot w_m}{10} + \in_1\right),\ \max_{H\subseteq G}\left\{\frac{w(E(H))}{\lfloor |V(H)|/2\rfloor} + \in_2\right\}\right\},$$

where $\in_1$ and $\in_2$ can be made arbitrarily small by taking t to be sufficiently small. But as t becomes smaller, finding an edge-coloring of G_t becomes more time consuming. The above procedure can be applied to the problem when speed-up is permitted and $\alpha(v)$ is any positive integer for all $v \in V(G)$ as follows. We construct a graph G_{1t} from G_t as follows: For each $v \in V(G_{1t})$, there are associated $\alpha(v)$ vertices $v(1), \ldots, v(\alpha(v))$ in $V(G_{1t})$. Each edge, among the $\mu(u,v)$ edges which join u and v in G_t is represented by an edge in G_{1t} which joins $u(i)$ to $v(j)$ for some i and j, $1 \le i \le \alpha(u)$ and $1 \le j \le \alpha(v)$. It can be verified that the representation in G_{1t} of edges of G_t can be done such that (i) $d_{G_{1t}}(v(i)) \le \lceil d_{G_t}(v)/\alpha(v)\rceil$ for $v \in V$ and $1 \le i \le \alpha(v)$; (ii) if $\mu_1(u(i), v(j))$ is the number of edges of G_{1t} which join $u(i)$ and $v(j)$, then $\mu_1(u(i), v(j)) \le \lceil \mu(u,v)/(\alpha(u)\cdot\alpha(v))\rceil$, where $\mu(u,v)$ is the number of edges of G_t which join u and v. It can be easily shown that one can develop a data transfer schedule based upon the above idea for which finishing time is $\le \frac{3}{2}w'_m (\le \frac{3}{2}$ optimum) if t is sufficiently small. The following procedure yields a slightly better performance and more efficient algorithm. We will describe this procedure but leave the proof to the reader as this case is similar to, in fact much simpler than the previous case studied in Section 3.2.

Algorithm 2.

Step 0.

We have graph $G(V,E)$, positive integers $\alpha(v)$ for all $v \in V$ and positive numbers $w(e)$ for all $e \in E$. Set $i \leftarrow 0$ and $t \leftarrow 0$. Compute $d_G(v)$ for all $v \in V$.

Step 1.

Set $i \leftarrow i+1$. Compute $w(v)$ (which is the sum of the weights of edges incident at v) and $w'(v) = w(v)/\alpha(v)$ for all $v \in V$. Let $w'_{m_i} = \max\{w'(v) | v \in V\}$.

Step 2.

(a) Compute $V_{1i} = \{v \in V | w'(v) = w'_{m_i}\}$. For each $e = (u,v) \in E(G)$ such that $w(e) \geq w'_{m_i}$, replace it by $\lceil \frac{w(e)+\in}{w'_{m_i}} \rceil = \mu(e)$ multiple edges with $\in$ any small positive number, make the weights of these edges all equal to $w(e)/\mu(e)$ and set $d_G(v) \leftarrow d_G(v) + \mu(e) - 1$ and $d_G(u) \leftarrow d_G(u) + \mu(e) - 1$.

(b) For each vertex v such that $d_G(v) < \alpha(v)$, pick an arbitrary edge $e = (u,v) \in E(G)$, replace it by $\big(\alpha(v) - d_G(v) + 1\big)$ multiple edges, make the weights of these edges all equal to $\frac{w(e)}{\alpha(v) - d_G(v) + 1}$, and set $d_G(v) \leftarrow \alpha(v)$ and $d_G(u) \leftarrow d_G(u) + \alpha(v) - d_G(v)$. Continue Step 2(b) until no such vertex exists.

(c) Compute subgraph H_i of G with proper labeling of H_i such that $\beta_{H_i} \leq \alpha(v)$ for all $v \in V$ and $\beta_{H_i}(v) = \alpha(v)$ for all $v \in V_{1i}$. (Multiple edges in H_i are permitted.) For $e = (u,w) \in E(G)$, let $\mu_i(e) = \sum\{\beta_{H_i}(e_1) | e_1 = (u,w) \in E(H_i)\}$. Compute

$$s_i = \min\Big\{\min\{\frac{w(e)}{\mu_i(e)} | e \in E(G)\}, \min\{\frac{\alpha(v) \cdot w'_{m_i} - w(v)}{\alpha(v) - \beta_{H_i}(v)} \mid v \in V - V_{1i}\}\Big\},$$

where $w(e)$ is the weight of edge e in G. For each $e \in E(G)$ if $e = (u,w) \in E(H_i)$, set $w(e) \leftarrow w(e) - \mu_i(e) \cdot s_i$ and if $w(e) = 0$, set $E \leftarrow E - \{e\}$, $d_G(u) \leftarrow d_G(u) - 1$, and $d_G(w) \leftarrow d_G(w) - 1$. If $d_G(v) = 0$, set $V \leftarrow V - \{v\}$.

Step 3.

(a) For each $e \in E(H_i)$ with $\beta_{H_i}(e) = 1$, start the data transfer corresponding to the edge e at time t for s_i seconds.

(b) A schedule for each edge $e \in E(H_i)$ with $\beta_{H_i}(e) = \frac{1}{2}$ is defined as follows: Let $e = (v_p, v_{p+1})$ belong to an odd cycle denoted by a set of edges (v_0, v_1), (v_1, v_2), $\ldots$, $(v_{2\ell}, v_0)$. Then the data transfer corresponding to the edge e become employed at time $t + p \cdot (s_i/2\ell)$ for $s_i/2\ell$ seconds and at time $t + \Big(\big(p - (-1)^q \cdot q\big) \bmod (2\ell+1)\Big) * s_i/2\ell$ for $s_i/2\ell$ seconds, for $q = 2, 3, \ldots, \ell$.

(c) Compute $t_i = s_i * |C_{is}| \;/\; (|C_{is}| - 1)$, where C_{is} represents the shortest odd cycle in H_i with $\beta_{H_i}(e_s) = \frac{1}{2}$ for all $e_s \in E(C_{is})$. Set $t \leftarrow t + t_i$.

(d) If $w'_{m_i} - s_i > 0$, then GO TO Step 1(a).

Step 4.

STOP.

Theorem 5. Let $G(V,E)$ be a communication graph and let $\alpha(v)$, $w(e)$, $w(v)$, $w'(v)$ and w'_m be as defined earlier. Then, we have:

(a) the above algorithm produces a correct schedule with finishing time

$$t^* \leq \ell_s/(\ell_s - 1) \cdot w'_m,$$

where ℓ_s represents the length of the shortest odd length cycle in G and this does not exceed $\ell_s/(\ell_s - 1)$ times the optimum finishing time and

(b) the time complexity of the above algorithm is

$$0\Big(|E| \cdot \big(|E| + \sum_{v \in V} \alpha(v)\big) \cdot \sum_{v \in V} \alpha(v)\Big).$$

Finally, we will describe another approach to our problem when speed-up is allowed. We will first construct a graph G_1 from a communication graph G as follows. For each vertex $v \in V(G)$, there are associated $\alpha(v)$ vertices $v(1), \cdots, v\big(\alpha(v)\big)$ in $V(G_1)$. The edge set $E(G_1)$ is defined such that if $e = (u,w) \in E(G)$, then $\alpha(u) \cdot \alpha(w)$ edges $\big(u(i), w(j)\big)$, for $1 \leq i \leq \alpha(u)$ and $1 \leq j \leq \alpha(w)$, are present in $E(G_1)$ with weights $w\big(u(i), w(j)\big) = w(e) \ / \ \big(\alpha(u) \cdot \alpha(w)\big)$, for $1 \leq i \leq \alpha(u)$ and $1 \leq j \leq \alpha(w)$. For each $v(i) \in V(G_1)$, we will set $\alpha\big(v(i)\big)$ to be equal to one. We note that $|V(G_1)| = \sum_{v \in V(G)} \alpha(v)$ and $|E(G_1)| \leq \left(\sum_{v \in V(G)} \alpha(v)\right)^2$. It is easy to see that one can use Algorithm 1 on G_1 to find an approximate solution to G when speed-up is allowed. Using this procedure it can be seen that the finishing time is at most $\ell_s/(\ell_s - 1) \cdot \max\{w(v)/\alpha(v) | v \in V\}$, where ℓ_s represents the length of the shortest odd cycle in G and the computational complexity is

$$0\left(|E_1|^2 \cdot \sum_{v \in V(G)} \alpha(v)\right) = 0\left(\left(\sum_{v \in V(G)} \alpha(v)\right)^5\right).$$

It can be seen that this approach although yields the same performance has higher computational complexity than Algorithm 2.

4. OPTIMUM SCHEDULING IN SPECIAL CASES

Choi and Hakimi [3] studied a version of our problem when a communication graph $G(V,E)$ is bipartite. They showed that the minimum finishing time is $\max\{\max\{w(e) \mid e \epsilon E\}, \max\{w(v)/\alpha(v) \mid v \epsilon V\}\}$ and provided an $0\big(|E|^2 \sum_{v \epsilon V} \alpha(v)\big)$ algorithm for finding a schedule with the minimum finishing time, when speed-up is not allowed. When speed-up is allowed, they also showed that the minimum finishing time is $\max\{w(v)/\alpha(v) \mid$

$v \epsilon V\}$ and provided an $0\Big(|E| \cdot (|E| + \sum_{v \epsilon V} \alpha(v)) \cdot \sum_{v \epsilon V} \alpha(v)\Big)$ algorithm for finding a schedule with the minimum finishing time.

Suppose a communication graph $G(V, E)$ is not bipartite but $\alpha(v)$ is even for all $v \epsilon V$. Then the problem when speed-up is allowed, can be reduced to the version of the problem studied in [3] as follows. We will construct a weighted bipartite graph $G_1(X, Y; E_1)$ from the communication graph $G(V, E)$. Let $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ where x_i and y_i correspond to vertex $v_i \in V = \{v_1, v_2, \ldots, v_n\}$. Two edges $e_1 = (x_i, y_j)$ and $e_2 = (x_j, y_i)$ are in E_1 if and only if $e = (v_i, v_j) \in E$ and the weights of both edges e_1 and e_2 are equal to $w(e)/2$. Let $\alpha(x_i) = \alpha(y_i) = \alpha(v_i)/2$, for $i = 1, 2, \ldots, n$. We note that the task corresponding to the data transfer between v_i and v_j in G, now corresponds to the edges (x_i, y_j) and (x_j, y_i) in G_1. Now if we apply Algorithm 2 in [3] to the bipartite graph G_1, we will find an optimum schedule which can immediately be translated into an optimum schedule for $G(V, E)$. It is easy to see that the finishing time of this schedule is $\max\{\frac{w(v)}{\alpha(v)} | v \in V\}$ and the time complexity of that algorithm is $0\Big(|E| \cdot (|E| + \sum_{v \in V} \alpha(v)) \cdot \sum_{v \in V} \alpha(v)\Big)$. It should be noted that we cannot use Algorithm 1 in [3] (which is designed to find an optimum schedule without speed-up for bipartite communication graphs) to find a schedule for $G(V, E)$ without speed-up. This is because Algorithm 1 may produce a schedule in which data transfers corresponding to the edges (x_i, y_j) and (x_j, y_i) are both engaged at the same instance of time. This schedule when translated into a schedule for $G(V, E)$ would imply the speed-up of task $(v_i, v_j) \in E$ by exactly a factor of two.

References

[1] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications*, American Elsevier Publishing Co., Inc., New York, 1976.

[2] L.G. Coffman, Jr., M.R. Garey, D.S. Johnson and A.S. La Paugh, "Scheduling File Transfers in a Distributed Network," *Proc. of 2nd Annual ACM Symposium on Principles of Distributed Computing*, 1983, pp. 254–266.

[3] H.-A. Choi and S.L. Hakimi, "Data Transfers in Networks," submitted for publication.

[4] I.S. Gopal, G. Bongiovanni, M.A. Bonuccelli, D.T. Tang and C.K. Wong, "An Optimal Switching Algorithm for Multibeam Satellite Systems with Variable Bandwidth Beams," *IEEE Trans. Comm.*, COM-30, 11, Nov. 1982, pp. 2475–2481.

[5] B. Hajek, "Link Schedules, Flows, and the Multichromatic Index of Graphs," *Proc. of Information Science and System*, Princeton, 1984.

[6] S. Fiorini and R.J. Wilson, *Edge-colourings of graphs*, Pitman, London/San Francisco/Melbourne, 1977.

[7] S.L. Hakimi and O. Kariv, "On a Generalization of Edge-Coloring in Graphs," preprint, Dept. of EECS, Northwestern University, Evanston, IL, (Sept. 1983). To appear in the *J. of Graph Theory.*

[8] C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization—Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[9] T. Nishizeki and K. Kashiwagi, "On the 1.1 Edge-Coloring of Multigraphs," preprint, Dept. of Electrical Communications, Tohoku University, Sendai, Japan, (April 1984).

DISTRIBUTED OPTIMAL SAMPLING WITH ASYNCHRONOUS COMMUNICATION

P.R.SRIKANTA KUMAR & G.K.YOUNG
Elec Engg& Comp Science
Northwestern University
Evanston,Illinois 60201

Elec,Comp,&Sys Engg
Rennselaer Poly Institute
Troy,NY 12181

ABSTRACT

The communication requirement for a two dimensional distributed optimization problem are considered.Decision makers have a finite set of alternatives,and face unknown distributions of rewards.It is shown that in n periods O(log n) messages are needed to achieve the optimal(maximum expected) time average of rewards.

INTRODUCTION

Distributed systems such as computer networks may be viewed in their operation as several decision makers(nodes or players) cooperating to acheive a common goal.In many instances the desired objective may be accomplished by coordinating or synchronizing the actions of the nodes.In large networks,however such coordination is often inefficient,even if possible. Furthermore loss of synchronzation may result in vastly inferior performance.In the absence of coordination some form of communication is necessary among the nodes.It is important to minimize the amount of communication,as it is expensive.
Here we consider a simple distributed decision system with two players.For each pair of actions chosen by the players they get a random reward(performance measure) which is the same for both.The distrbution of rewards is initially unknown.Let d be the maximum expected reward possible if the reward distrbutions are known,and if both cooperate.We consider procedures where each player selects his action,sequentially over time,based on his past information and on that gathered communicating with the other via elementary messages. It is shown that communication required is O(log n) to recieve a time average reward equal to d(ie.,accumulated errors due to learning and distribution is bounded).Communication is asynchronous.By this we mean that the instants at which messages are sent are not synchronized,instead they are independly chosen by the nodes.

PROBLEM AND MAIN RESULT

Consider two players A and B.Each has a finite number of actions.Let X(i,j) be the random reward(payoff) if A selects his action i,and B selects his action j.Let d(i,j)=E X(i,j) , be the expectation.Let

$$d = \max_{i,j} d(i,j)$$

The players ,over time,select their actions based on the past observations, via a sequential procedure. Let n=0,1,2...be time instants. If at n players choose (i,j), the reward Z(n) recieved at n is a realization of X(i,j). A procedure is optimal(or is an optimal sampling plan)if the time average of rewards equals d ie.,

$$\lim_{n \to \infty} 1/n \sum_{k=1}^{n} Z(k) = d.$$

In a distributed context, a player, more often than not, knows only about his actions and the rewards recieved, and not the actions of the other. In case of complete ignorance about other players there is no optimal or ϵ-optimal procedure. Here we allow communication to take place via messages of the form (action, time) which tells the reciever the specific action chosen by the sender at the specified time. Messages may reach after a bounded delay. The players independently select their communication instants. We address the question of the minimal number of messages required for optimality.

Remarks: For the case of all players knowing everything about the others, Robbins[1] has devised an optimal sampling plan. In this case the problem is equivalent to a one player situation. Following Robbins' scheme it is possible to have an optimal procedure where no communication is required, the players have to agree upon a synchronized sequence of times at which they sample specific actions. However, it is easy construct examples where small errors in the synhronized sequence result in convergence far away from optimal.

Procedure: Each player has a schedule of communication times. If k is such a time of A, then A sends message (a(k),k). a(k) is the action chosen by A at k. After recieving this, B has the complete event (a(k),b(k),Z(k)). From such information both players arrive at sample mean estimates of X(i,j). Estimates are updated whenever a message arrives. They are, at any time, different in general for the two players. A preferred action for a player, at any time, is that correspoding to his highest current estmate. At each time players pick their preferred action with prob. 1-ϵ, and from the others(nonpref.)with prob $\epsilon > 0$.

Let KA(n) and KB(n) be the total number of communication times for A and B respectively.

Theorem: If KA(n) and KB(n) are both O(log n), and if first and second moments of X(i,j) exist, then with the above described procedure $(1/n)\sum_{k=1}^{n} Z(k) \to d$ as $n \to \infty$ and $\epsilon \to 0$. (Detailed proof may be found in [2].)

REFERENCES

1 .Robbins .H."Some aspects of sequential design of experiments"Bull Amer Math Soc.,Vol58,pp527-535 1952.

2 .Srikanta Kumar.P.R.,and G.K.Young.,..techreport Northwestern university.

A NONCLASSICAL WRLS ALGORITHM

TAT CHIU LUK
Baldwin Technology Corporation
5118 S. Dansher Rd.
Countryside, IL 60525
(312) 579-1600

JOHN R. DELLER, JR.
Department of Electrical and Computer Engineering
and The Center for Research on Hearing Speech and Language
Northeastern University
Boston, MA 02115

This work was performed in part at the
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago

ABSTRACT

The conventional weighted recursive least squares (WRLS) algorithm [1] is widely used in many control, estimation, and identification problems. In contrast to "batch" solutions, this method offers a built in check for convergence, as well as the ability to conveniently examine and weight incoming data sequentially. Whereas statisticians and systems practicioners have long been aware of the advantages of the weighted recursive solution. This method has ignored by speech (and other signal) processing engineers due in large part to its relatively poor computational efficiency and also due to a lack of interest in weighted solutions in general which naturally motivate recursive methods. Recent work by the authors [2-7] has engendered an interest in the application of WRLS to speech analysis and we have sought measures for making the recursion more efficient. In this paper, we develop and study a nonclassical algorithm for RLS estimation which produces a theoretically equivalent result to the classical method, and which retains all of its advantages, while being computationally more efficient and having other useful analytical properties as well. This technique is similar conceptually to the Givens' transformation (GT), a method conventionally applied to unweighted, batch type solutions.

THEORETICAL BASES

Since our motivating interest is in time series analysis - in particular, the linear predictive coding of speech - we begin by posing the problem as one of generalized (weighted) linear prediction :

Problem 1 (P1). Derive the coefficients, $\hat{a}_i(N)$, of the M order linear predictor of the sequence $x(n)$ on n [1,N],

$$\hat{x}_N(n) = \sum_{i=1}^{M} \hat{a}_i(N)x(n-i)$$

$$= \underline{\hat{a}}^T(N)\ \underline{x}(n) \tag{1}$$

which minimizes

$$J(N) = [1/N] \sum_{n=1}^{N} q_n\ \xi_N^2(n) \tag{2}$$

where, $\xi_N(n)$ is the prediction error sequence,

$$\xi_N(n) = \hat{x}_N(n) - x(n) \tag{3}$$

and q_n is some set of weights.

The solution is the set of well-known "normal" [8] or "covariance" [9] equations, generalized to include weights.

In the more classical least squares theory, this problem is posed as an overdetermined system of equations subject to a weighted norm minimization [1]. Here we pose an unconventional, but nevertheless equivalent, variation on this problem consisting of a "weighted" system of equations, subject to an unweighted norm minimization. (This is done to permit a more elegant integration of the weights into the developments to follow.)

Problem 2 (P2). Define $\underline{X}(n)$ to be the Mxn matrix with i^{th} column $\underline{x}(i)$, $\underline{x}'(n)$ as the vector $[x(1)\ x(2)\ \ldots\ x(n)]^T$, and also,

$$\underline{Y}(n) = \sqrt{\underline{Q}(n)}\underline{X}(n); \qquad \underline{y}'(n) = \sqrt{\underline{Q}(n)}\underline{x}'(n) \tag{4}$$

where, $\sqrt{\underline{Q}(n)} = \mathrm{diag}(\sqrt{q_1}\ \sqrt{q_2}\ \ldots\ \sqrt{q_n})$.

Consider the overdetermined set of equations

$$\underline{Y}^T(N)\underline{a} = \underline{y}'(N) \tag{5}$$

where $\underline{a}$ is unknown. Find the estimate of $\underline{a}$, say $\hat{\underline{a}}(N)$, which minimizes

$$J(N) = \|\underline{y}'(N) - \underline{Y}^T(N)\hat{\underline{a}}(N)\|^2 \tag{6}$$

The well-known batch solution to (P2) takes the form

$$\hat{\underline{a}}(N) = Y^{+}(N)\underline{y}'(N) = (\underline{Y}(N)\underline{Y}^{T}(N))^{-1}\ \underline{Y}(N)\underline{y}'(N) \qquad (7)$$

where $\underline{Y}^{+}(N)$ is the generalized inverse of $\underline{Y}^{T}(N)$ [10,p.19], and, with a little effort, one can show that this solution is tantamount to the "normal equation" solution of (P1). Also note that the standard recursive solution to (P2) (and, hence, to (P1)) can be found in many sources, e.g.[1,Ch.3].

Now let us consider the system of equations (5) at a general time, n

$$\underline{Y}^{T}(n)\underline{a} = \underline{y}'(n) \qquad (8)$$

The following are prerequisite to the development of the new method:

Propositions

(i) Any number of "null equations" of the form $\underline{0}^{T}\ \underline{a} = 0$, where $\underline{0}$ represents the zero vector, may be added to the system (8) without effecting the least square (LS) solution of (7)

(ii) An orthonormal transformation of the system (8) will not effect the solution, and, further,

(iii) If an orthonomal transformation on (8) results in the form

$$[\underline{T}^{T}(n) \mid \underline{0}^{T}]^{T}\ \underline{a} = [\underline{d}_1^{T}(n) \mid \underline{d}_2^{T}(n)]^{T} \qquad (9)$$

where $\underline{T}(n)$ is upper trianglular, and $\underline{0}$ is an (N-M)xM zero matrix, and $\underline{d}_1(n)$ is dimension M, the LS solution at time n, is the solution of

$$\underline{T}(n)\ \hat{\underline{a}}(n) = \underline{d}_1(n) \qquad (10)$$

(Note that the solution of (10) is trivial (achieved by back-substitution [10,p.59]) compared with that of (7).)

Propositions (i) and (ii) are easily demonstrated, while (iii) is shown as follows :

The LS solution to the system of equations (9) (which is also the LS solution to (8) according to Prop. (ii) is given by (see (7))

$$([\underline{T}^T(n) | \underline{0}^T] \; [\underline{T}^T(n) | \underline{0}^T]^T)^{-1} \; [\underline{T}^T(n) | \underline{0}] \; [\underline{d}^T{}_1(n) | \underline{d}_2^T(n)]^T$$

$$= (\underline{T}^T(n) \; \underline{T}(n)) \; \underline{T}^T(n) \; \underline{d}_1(n)$$

$$= \underline{T}^{-1}(n) \; \underline{d}_1(n) \qquad (11)$$

which is the solution to (10).

Also requisite is the introduction of plane rotation operators (see, e.g. [11,p.321]). An LxL plane rotation matrix $\underline{R}_{j,l}(\theta)$, differs from the identity matrix only in the elements

$$r_{jj} = r_{ll} = \cos\theta$$

$$r_{jl} = -r_{lj} = \sin\theta \qquad (12)$$

where, if $\underline{R}_{j,l}(\theta)$ is to operate on, say $\underline{B}$, then θ is chosen as the desired "angle of rotation" of the plane spanned by the j^{th} and l^{th} variables of $\underline{B}$, i.e., by the vectors $[b_{jj} \;\; b_{lj}]^T$ and $[b_{jl} \;\; b_{ll}]^T$. In the present application, we will always choose θ in such a way that the element b_{lj} becomes zero under the rotation. From the definition of $\underline{R}_{j,l}(\theta)$, it is seen that this requires

$$-b_{jj} \sin\theta + b_{lj} \cos\theta = 0 \Rightarrow \tan\theta = b_{lj}/b_{jj} \qquad (13)$$

We are now prepared to develop the formal batch solution of (5) based on these precepts. The method given here is essentially equivalent to the development by Nash [8,Sect.4.2] with a different initialization process which is more amenable to the recursion. (Note: we avoid the use of the term "Givens' transformation" which is used by Nash to describe the rotations, as this term is more commonly associated with an algorithm used to reduce a symmetric matrix to tridiagonal form.) Let us add M null equations (Prop. (i)) at the top of (5), denoting the new system

$$\underline{B}\;\underline{a} = \underline{c}\;; \quad \underline{B} = \begin{bmatrix} \underline{0} \\ \\ \underline{Y}^T(N) \end{bmatrix}, \quad \underline{c} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ \underline{y}'(N) \end{bmatrix} \qquad (14)$$

We now apply the following sequence of rotations to the system (14)

$\underline{R}_{1,M+1}$, $\underline{R}_{2,M+1}$, ..., $\underline{R}_{M,M+1}$ (row M+1)
then $\underline{R}_{1,M+2}$, $\underline{R}_{2,M+2}$, ..., $\underline{R}_{M,M+2}$ (row M+2)
:
:
then $\underline{R}_{1,M+N}$, $\underline{R}_{2,M+N}$, ..., $\underline{R}_{M,M+N}$ (row M+N) (15)

In each case, θ is computed according to (13). The following are to be observed.

Observations

(i) If applied in the prescribed order, the result of these MN rotations (on $\underline{B}$) is an upper triangular form with N rows of zeros below.

(ii) Further, the upper triangular form has been produced by an orthonormal operation on the system (since each $\underline{R}_{jl}$ is orthonormal); hence, the result is in form (9) and $\hat{\underline{a}}(N)$ may be obtained by solution of (10).

(iii) The application of $\underline{R}_{j,l}$ effects only rows j and l of $\underline{B}$ and $\underline{c}$, and in fact, only columns j, j+1, ..., M of $\underline{B}$ (those elements in columns 1,2, ..., j-1 were previously set to zero and remain so under the rotation).

(iv) After the l^{th} equation (row in $\underline{B}$ and $\underline{c}$) has been set to zero by the corresponding sequence of rotations, this row plays no further role in the solution.

SEQUENTIAL ALGORITHM

The Algorithm

These observations make possible a modified procedure which is "sequential" in the sense that it includes each equation (representing a time n) "as it becomes available." Since at any time, only M+1 rows of $\underline{B}$ and $\underline{c}$ contribute to the process, we allocate only (M+1)xM storage locations to $\underline{B}$ and (M+1)x1 to $\underline{c}$. In fact, these can be concatenated into a "working matrix", $\underline{W}$, of dimension (M+1)x(M+1). By $\underline{W}(n)$ we will mean the working matrix at time n, pre-rotation, and by $\underline{W}'(n)$, post-rotation. The elements are w_{ij} and w'_{ij}, respectively.

Initially, all elements of $\underline{W}$ should be zero.

Step 0:

$$\underline{W}(0) = \left[\underline{0}_{(M+1)\times M} \;\middle|\; \underline{0}_{(M+1)\times 1}\right]$$

For n=1,2, ...,N

Step 1 : Replace the last row of $\underline{W}'(n-1)$ by

$$\begin{bmatrix} \underline{y}'(n) \\ \\ \sqrt{q_n}\ x(n) \end{bmatrix}^T = \begin{bmatrix} \sqrt{q_n}\ \underline{x}(n) \\ \\ \sqrt{q_n}\ x(n) \end{bmatrix}^T$$

to form $\underline{W}(n)$. (this amounts to discarding the henceforth unnecessary equation resulting from the previous rotation (Obs. (iv)).

Step 2 : Rotate row (M+1) into the upper matrix. (Formally $\underline{R}_{j,l}$ is now (M+1)x(M+1), however, due to Obs. (iii), the calculation is achieved simply by letting

$$w'_{jk} = w_{jk} \cos \theta + w_{M+1,k} \sin \theta$$

$$w'_{M+1,k} = -w_{jk} \sin \theta + w_{M+1,k} \sin \theta$$

for k =j, j+1, ..., M+1; and replacing no other elements. In each case θ is computed as in (13).

Step 3 : An upper triangular form occurs in W'(n) which may optionally be used to solve for $\underline{\hat{a}}(n)$ (Obs. (i), (ii)). (This may be used to check for convergence.)

Step 4 : Return to step 1 with next n if n ≠ N or covergence criterion is not met in step 3.

Computational Efficiency

The computational efficiency of the new sequential algorithm is contrasted with that of conventional WRLS in Table 1. More detail is given in the work of Luk [3]. In computing these numbers, the operations to achieve the solutions $\underline{\hat{a}}(n)$ using batch substitution have been ignored. Each such computation requires M multiplications and M(M+1)/2 additions.

Using the number of multiplies as a figure of merit, Fig. 1 shows the relative computational loads of the two methods as a functions of model order, M. It is clear that the new method yields a computational advantage over the conventional for model orders exceding five. This advantage becomes larger with M.

Table 1. Computational efficiency of the new algorithm compared with standard WRLS.

	Number of Multiplications per n	Number of Additions per n
New Algorithm	$2M^2 + 11M + 2$	$M^2 + 3M + 2$
Conventional Algorithm	$2M^2 + 5M + 3$	$2.5M^2 + .5M - 1$

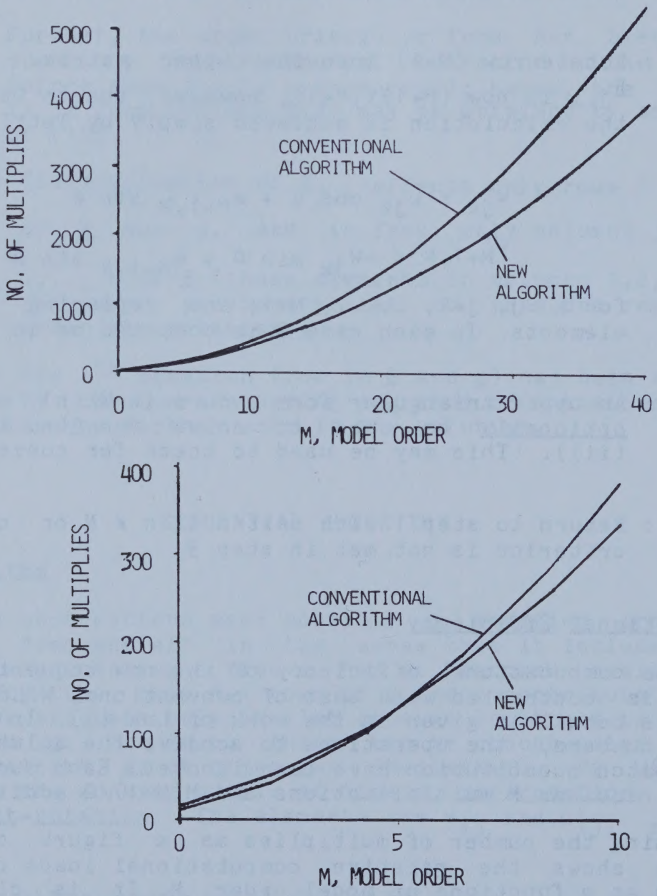


Figure 1.(a) Number of operations vs. model order for the classical and new algorithms. (b) represents a "blowup" of the low model order portion of (a).

Other Useful Features

In the motivating batch solution of Section 2, the final result contains a heretofore unused vector, $\underline{d}_2(N)$, in the lower paer of the right-hand vector. It is useful to note that $\underline{d}_2(N)$ has an interesting interpretation as the vector whose norm equals the minimal cost function, J(N), of (2) or (6). This is demonstrated as follows:

From (6) it is easy to show that

$$J(N) = \underline{y}'^T(N)\underline{y}'(N) - \underline{y}'^T(N)\underline{Y}^T(N)\underline{\hat{a}}(N) - \underline{\hat{a}}(N)\underline{Y}(N)\underline{y}'(N) + \underline{\hat{a}}^T(N)\underline{Y}(N)\underline{Y}^T(N)\underline{\hat{a}}(N) \tag{16}$$

Noting (5), however, the last two terms add to zero so that

$$J(N) = \underline{y}'^T(N)\underline{y}'(N) - \underline{y}'^T(N)\underline{Y}^T(N)\underline{\hat{a}}(N) \tag{17}$$

Letting $\underline{R}$ stand for the global transformation that takes place on the system of equations, we can write

$$J(N) = \underline{y}'^T(N)\underline{R}^T\underline{R}\underline{y}'(N) - \underline{y}'^T(N)\underline{R}^T \begin{bmatrix} \underline{T}(N) \\ \underline{0} \end{bmatrix} \underline{\hat{a}}(N) \tag{18}$$

Since,

$$\underline{R}^T \underline{R} = \underline{I} \quad \text{and} \quad \begin{bmatrix} \underline{T}(N) \\ \underline{0} \end{bmatrix} \underline{R}^T = \underline{Y}^T(N) \tag{19}$$

Therefore

$$J(N) = \begin{bmatrix} \underline{d}_1(N) \\ \underline{d}_2(N) \end{bmatrix}^T \begin{bmatrix} \underline{d}_1(N) \\ \underline{d}_2(N) \end{bmatrix} - \begin{bmatrix} \underline{d}_1(N) \\ \underline{d}_2(N) \end{bmatrix}^T \begin{bmatrix} \underline{T}(N) \\ \underline{0} \end{bmatrix} \underline{\hat{a}}(N)$$

$$\begin{aligned} &= \underline{d}_1^T(N)\underline{d}_1(N) + \underline{d}_2^T(N)\underline{d}_2(N) - \underline{d}_1^T(N)\underline{d}_1(N) \\ &= \underline{d}_2^T(N)\underline{d}_2(N) \end{aligned} \tag{20}$$

Although we will not prove it here, it is believable in this light, that the i^{th} element of $\underline{d}_2(N)$, say $d_{2,i}(N)$, is the prediction error at time i based on the LS estimate $\hat{\underline{a}}(N)$. We have called this number $\xi_N(i)$ in (3). With some further thought, therefore, one can believe that in the new method, the element remaining in the (M+1, M+1) location of the working matrix after rotation ($w'_{M+1,M+1}$) at time n, is the number $\xi_n(n)$, the "prediction" of the current time based on the current parameter vector, $\hat{\underline{a}}(n)$. This number is useful in certain applications (see [2-7]).

Note that a second useful feature is that a previously incorporated equation (observation at time n) is easily removed from the estimate by simple "back-rotations," i.e., by applying the sequence (following (15)) $\underline{R}^T_{i,n+M}$, $i=1,2,\ldots,M$ to the working matrix. This feature can also be useful in certain data selection applications and is unavailable with the conventional technique.

CONCLUSIONS

A nonclassical WRLS algorithm based on a novel reorganization of a fundamentally "batch" type solution has been developed. The method enjoys a computational advantage over the classical WRLS method which increases with the order of the system. Unlike the classical method, the new method also computationally straightforward and simple to program. Like the classical method, the new algorithm may treat points temporally sequentially (or in fact, in any order), it involves no matrix inversions, and provides a convenient check for convergence of the estimator at each n.

We note finally the relationship of this new method to well known numerical analysis methods. Plane rotation techniques are routinely suggested as means of simplifying simultaneous equation solutions or eigenvalue computations. Notable examples are the Givens' Reduction and the Householder Transformation [11,Sect.10.5,10.6]. Such approaches (when applied to nonsymmetric matrices such as those in the present problem) will at best reduce the matrix to lower Hessenberg form, are typically used in a "batch" form, and do not address the problem of weighting. The technique suggested here has extended and improved upon these approaches in these three important ways.

REFERENCES

[1] T. C. Hsia, System Identification, Heath, Lexington, Massachusetts, 1977.

[2] J. P. Laebens and J. R. Deller, Jr., "'SISSI' - A Silent Input Selective Sequential Identifier for AR Systems", Conference Proceedings : Neuvieme Colloque sur le Traitement du Signal et ses Applications, Vol. 2, pp.1041-1045, 1983.

[3] T. C. Luk, "Fast Optimal Identification of AR Systems with a Class of Correlated Inputs : Application to Glottal Deconvolution" (Ph.D. Dissertation), Illinois Institute of Technology, Chicago, 1985.

[4] J. R. Deller, Jr. and T. C. Luk, "An Optimal Silent Input Selective Sequential Identifier for AR Systems," Proc. IASTED Int. Conf. on Applied Ident., Model., and Simul., pp.48-51, New Orleans 1984.

[5] T. C. Luk and J. R. Deller, Jr., "Sequential Closed Phase Glottal Inverse Filtering," Proc. Acoustic Soc. Amer. Fall Meeting 1985, Nashville (Abstract).

[6] J. R. Deller, Jr. and T. C. Luk, "Sequential Closed-Phase Glottal Inverse Filtering," in preparation.

[7] J. R. Deller, Jr. and T. C. Luk, "Set Membership Set Theory Applied to the Linear Predictive Coding of Speech," in preparation.

[8] J. D. Markel and A. H. Gray, Linear Prediction of Speech, Springer-Verlag, Berlin, Heidelberg, New York, 1976.

[9] G. E. Box and G. H. Jenkins, Time Series Analysis: Forcasting and Control, Holden Day, San Francisco, 1976.

[10] J.C. Nash, Compact Numerical Methods For Computers, Linear Algebra and Function Minimization, John Wiley & Son, New York, N.Y., 1979.

[11] B. Nobel, Applied Linear Algerbra, Prentice Hall, Englewood Cliffs, N.J., 1969.

A FIXED - POINT COMPUTATION FOR LINEAR PHASE LATTICE FILTERING

BY **DAVID C. FARDEN** AND **JEROME R. BELLEGARDA**
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ROCHESTER
ROCHESTER, NY 14627

ABSTRACT

Toeplitz sets of linear equations arise in a variety of applications, including MMSE FIR filter design, spectral analysis and system identification. In many real-time instances, a fixed-point computation is required. This paper defines a new parameterization for the solution to the Toeplitz system which results from a MMSE estimation problem for the design of a *linear phase* FIR filter. The parameterization is exploited to derive a *fixed-point computation* of the solution, and leads to a new lattice/ladder FIR realization, which is appropriate for the implementation of any linear phase FIR filter. With respect to a direct form realization, the new structure has improved numerical properties and is better suited for adaptive filtering, which enables a stochastic gradient algorithm to be used efficiently.

I. INTRODUCTION

Because FIR (finite duration unit pulse response) digital filters are always stable, have low word length requirements, and can be designed to have linear phase, they have found frequent application [1]: it is therefore important to minimize their cost of implementation. One technique for designing an FIR filter is to pose the design problem as a MMSE (minimum mean square error) estimation problem [2], which requires the solution to a set of Toeplitz linear equations. Toeplitz systems of equations arise in a variety of other applications, including spectral analysis and system identification. In many real-time instances, a fixed-point computation is required. This paper considers a MMSE estimation problem for the design of a *linear phase* FIR filter, and presents a *fixed-point computation* of a parameterized solution to the resulting Toeplitz system. The parameterization leads to a new lattice/ladder FIR realization, which is appropriate for the implementation of any linear phase FIR filter. With respect to a direct form realization, the new structure has improved numerical properties and is better suited for adaptive linear phase filtering.

The paper is organized as follows. The rest of the introduction presents the MMSE estimation problem for the design of a linear phase FIR filter, the notation for the corresponding set of Toeplitz linear equations, and some background facts on lattice structures, which play a crucial role in the new FIR realization. In Section II, the recursive algorithm is derived, and in the next section, the resulting new FIR structure is presented. Some computational considerations and a flowchart for implementing the algorithm are given in Section IV. Finally, a procedure is developed in Section V for using the new linear phase lattice FIR structure as an adaptive filter.

Problem Statement

Let $\{y_n\}$ and $\{d_n\}$ be jointly wide-sense stationary sequences of real zero mean random variables, with correlation sequences defined by $\rho_i = E[y_n y_{n-i}]$ and $\sigma_i = E[d_n y_{n-i}]$, where $E[\cdot]$ denotes statistical expectation. Let $Y_{m,n} \in \mathbf{R}^m$ represent the $m \times 1$ data vector: $Y_{m,n} = [y_n \; y_{n-1} \cdots y_{n-m+1}]'$, where $'$ denotes vector transpose. We define the problem as follows: find the best linear MMSE estimate $(\hat{d}_{m,n})$ of $d_{m,n}$ (a shifted version of d_n), using the observed data $Y_{m,n}$; i.e. choose $\gamma_m \in \mathbf{R}^m$ to minimize $E[e^2_{m,n}]$, where $e_{m,n} = d_{m,n} - \hat{d}_{m,n}$ and $\hat{d}_{m,n} = -\gamma'_m Y_{m,n}$. From the orthogonality principle, the minimizing γ_m is found to be a solution to:

$$R_m \gamma_m = -s_m , \tag{1}$$

where R_m is the $m \times m$ Toeplitz matrix with $(i,j)^{\text{th}}$ element ρ_{j-i}, and s_m is the $m \times 1$ vector $E[d_{m,n} Y_{m,n}]$. The resulting MMSE is $\xi_m = E[d_n^2] + s'_m \gamma_m$.

The preceding problem statement is valid for any positive integer m: the solution γ_m to (1) represents the delta response of a length m FIR filter. If $\gamma_m = \check{\gamma}_m$, or if $\gamma_m = -\check{\gamma}_m$, where $\check{}$ denotes reversed ordering, (i.e. $\check{\gamma}_m$ is γ_m with all the elements placed in reversed order), then γ_m represents a *linear phase* FIR filter [1]. It is easily shown [2]-[4] that if $s_m = \check{s}_m$ then the solution to (1) satisfies $\gamma_m = \check{\gamma}_m$; similarly, it can be shown that if $s_m = -\check{s}_m$ then $\gamma_m = -\check{\gamma}_m$.

As an example, consider an observation model $y(t) = d(t) + \eta(t)$, where $\{d(\cdot)\}$ and $\{\eta(\cdot)\}$ are real, zero mean, jointly wide-sense stationary random processes with $E[d(t+\tau)\,\eta(t)] = 0$ for all real τ. Let $y_n = y(nT)$ and $d_n = d(nT)$. Choosing $d_{m,n} = d([n-(m-1)/2]T)$ (see [3]) in this case yields $s_m = \check{s}_m$, and hence $\gamma_m = \check{\gamma}_m$. Similarly, assuming that $d(\cdot)$ is mean square differentiable, choosing $d_{m,n} = \frac{d}{dt}[d(t)]_{t=[n-(m-1)/2]T}$ yields $s_m = -\check{s}_m$ and $\gamma_m = -\check{\gamma}_m$.

We assume in the following that $d_{m,n} = d_{n-(m-1)/2}$ so that $s_m = [\sigma_{-(m-1)/2}\ \sigma_{-(m-3)/2} \cdots \sigma_{(m-1)/2}]'$. We further assume that $\sigma_{-i} = \sigma_i$ to ensure that $s_m = \breve{s}_m$. The case $s_m = -\breve{s}_m$ follows a similar development. We seek an efficient solution to (1) when $m = M$, M being the resulting filter order. The algorithm presented in Section II consists of efficient order update equations for a *parameterized solution* of (1).

Preliminaries

Two auxiliary quantities provide useful intermediate results for solving (1) efficiently. Define the forward and backward prediction errors as $f_{m,n} = y_n + \alpha_m' Y_{m,n-1}$ and $b_{m,n} = y_{n-m} + \breve{\alpha}_m' Y_{m,n}$ respectively, where $\alpha_m \in \mathbf{R}^m$ is the value which minimizes $E[f_{m,n}^2]$ $(= E[b_{m,n}^2])$ with resulting MMSE ε_m. It is easy to show that $\varepsilon_m = \rho_0 + r_m' \alpha_m$, where $r_m = [\rho_1\ \rho_2 \cdots \rho_m]'$. The well-known Levinson–Durbin recursions [5]:

$$\alpha_{m+1} = \begin{bmatrix} \alpha_m \\ 0 \end{bmatrix} + k_m \begin{bmatrix} \breve{\alpha}_m \\ 1 \end{bmatrix} \tag{2}$$

and

$$\varepsilon_{m+1} = (1 - k_m^2)\, \varepsilon_m , \tag{3}$$

provide an order recursion for α_m and ε_m, where the so-called reflection coefficient k_m is given by:

$$k_m = -\frac{(\rho_{m+1} + r_m' \breve{\alpha}_m)}{\varepsilon_m} . \tag{4}$$

The resulting set of reflection coefficients $\{k_m\}_{m=0}^{M-1}$ provides a convenient parameterization of the M^{th} order prediction vector α_M and enables $\{f_{m,n}\}_{m=0}^{M-1}$ and $\{b_{m,n}\}_{m=0}^{M-1}$ to be obtained with a lattice structure, via the recursions:

$$f_{m+1,n} = f_{m,n} + k_m b_{m,n-1} , \tag{5}$$

$$b_{m+1,n} = k_m f_{m,n} + b_{m,n-1} . \tag{6}$$

We want to extend these results to provide a corresponding parameterization for the linear phase estimator.

II. RECURSIVE ALGORITHM

In [6], Le Roux and Gueguen define the new variable:

$$\varepsilon_{m,i} = E[f_{m,n}\, y_{n-i}] \tag{7}$$

and show that (note that $\varepsilon_m = \varepsilon_{m,0}$):

$$k_m = -\frac{\varepsilon_{m,m+1}}{\varepsilon_m}, \tag{8}$$

$$\varepsilon_{m,i} = 0, \qquad \text{for } i = 1, 2, \ldots, m, \tag{9}$$

$$\varepsilon_{m+1,i} = \varepsilon_{m,i} + k_m \varepsilon_{m,m+1-i}, \qquad \text{for all integer } i. \tag{10}$$

To obtain the set of reflection coefficients $\{k_m\}_{m=0}^{M-1}$, (10) must be computed for $m = 0, 1, \ldots, M-1$ with $i = 1-M+m,\ 2-M+m, \ldots, M$ and initial conditions $\varepsilon_{0,i} = \rho_i$. However, from (9), it is seen that the computation of (10) can be omitted for $i = 1, 2, \ldots, m+1$, which significantly reduces the computational load. Note that M must be known at the start of the procedure.

This approach permits the recursion on the abstract variable α_m in the Levinson–Durbin algorithm to be replaced by the above order recursion on the physical quantity $\varepsilon_{m,i}$. The advantage of doing so is that $\varepsilon_{m,i}^2 \le \rho_0 \varepsilon_m$ from the Cauchy–Schwarz inequality. This bound on the variable used in the recursion readily permits a *fixed-point implementation* to compute the reflection coefficients [6].

Inspired by the above results, we now present a new parameterization to replace γ_m as well, and derive a fixed point computation for the MMSE linear phase FIR filtering problem (1). Let us introduce the new variable:

$$\varsigma_{m,i} = E\left[e_{m,n}\, y_{n-i}\right], \tag{11}$$

which satisfies (Cauchy–Schwarz inequality) $\varsigma_{m,i}^2 \le \rho_0 \xi_m$. The orthogonality principle — or, equivalently, (1) — implies that:

$$\varsigma_{m,i} = 0, \qquad \text{for } i = 0, 1, \ldots, m-1. \tag{12}$$

We note that:

$$s_{m+2} = \begin{bmatrix} \sigma_{(m+1)/2} \\ s_m \\ \sigma_{(m+1)/2} \end{bmatrix}, \qquad m \ge 0, \tag{13}$$

where s_0 is null. Furthermore, γ_m needs to be computed only for even (odd) values of m if an even (odd) order solution is desired. It has been shown [3], [7] that an order update for γ_m can be expressed as:

$$\gamma_{m+2} = \begin{bmatrix} 0 \\ \gamma_m \\ 0 \end{bmatrix} - \frac{\sigma_{(m+1)/2} + r'_m \gamma_m}{\varepsilon_{m+1}} \left(\begin{bmatrix} 1 \\ \alpha_{m+1} \end{bmatrix} + \begin{bmatrix} \tilde{\alpha}_{m+1} \\ 1 \end{bmatrix} \right). \tag{14}$$

We use (11) along with (14) to obtain the recursion:

$$\zeta_{m+2,i} = \zeta_{m,i-1} - c_{m+1}\left(\varepsilon_{m+1,i} + \varepsilon_{m+1,m+1-i}\right), \tag{15}$$

where:

$$c_{m+1} = \frac{\zeta_{m,m}}{\varepsilon_{m+1}}. \tag{16}$$

As for γ_m, the variables $\zeta_{m+2,i}$ and c_{m+1} need to be computed only for even (odd) values of m if M is even (odd). Some more computation shows that:

$$\hat{d}_{m+2,n} = \hat{d}_{m,n-1} + c_{m+1}(f_{m+1,n} + b_{m+1,n}) \triangleq \hat{d}_{m,n-1} + c_{m+1}\,\psi_{m+1,n}. \tag{17}$$

Consequently, the recursion (14) on γ_m can be eliminated: the only variables which are needed for the algorithm are now $\zeta_{m,i}$ and $\varepsilon_{m,i}$. The coefficients $\{c_m\}_{m=0}^{M-1}$ and $\{k_m\}_{m=0}^{M-2}$ represent a *parameterized solution* to (1) at stage M, i.e. to $R_M \gamma_M = -s_M$.

To obtain this solution, the recursion (15) must be computed (i) if M is even, for $m = 0, 2, 4, \ldots, M-4$ with initial conditions $\zeta_{0,i} = \sigma_{i+\frac{1}{2}}$, and (ii) if M is odd, for $m = 1, 3, 5, \ldots, M-4$ with initial conditions $\zeta_{1,i} = \sigma_i - \frac{\sigma_0}{\rho_0}\rho_i$ and $c_0 = \frac{\sigma_0}{2\rho_0}$. For each required value of m, $\zeta_{m+2,i}$ in (15) is needed for $i = \frac{m-M}{2}+1, \frac{m-M}{2}+2, \ldots, \frac{m+M}{2}$. However, since $\gamma_m = \tilde{\gamma}_m$, we have $\zeta_{m,i} = \zeta_{m,m-1-i}$; and from (12), $\zeta_{m+2,i} = 0$ for $i = 0, 1, \ldots, m+1$; hence the required computation of (15) can be reduced to values of $i \in \left\{m+2, m+3, \ldots, \frac{m+M}{2}\right\}$. The computational count for $\{\zeta_{m,i}\}$ and $\{c_m\}$ can then be evaluated to a total of $\frac{1}{8}M(M+2)$. The parameterized solution to (1) with $m = M$ thus requires approximately $1.125\,M^2$ multiplications compared with $1.25\,M^2$ required for computing γ_M using the algorithm in [3]. We note that other algorithms are known to obtain the solution γ_M in $O(M \log^2 M)$ operations [8], [9]; however, the large leading coefficient involved in the $O(\cdot)$ notation (about 40 according to [8]) and the associated overhead operations apparently make these algorithms require a very large M to be competitive with the one presented here. In addition, the algorithm derived here is directly suitable for fixed-point implementation, as will be further explained in Section IV.

We now show that a recursion similar to (3) can be obtained for the minimum mean square estimation error ξ_m, again without explicit use of γ_m. Noting that $\hat{d}_{m+2,n}$ and $\hat{d}_{m,n-1}$ are $(m+2)^{\text{nd}}$ and m^{th} order MMSE estimates, respectively, of $d_{n-(m+1)/2}$, it follows that:

$$e_{m+2,n} = e_{m,n-1} - c_{m+1}\,\psi_{m+1,n}, \tag{18}$$

where c_{m+1} is the value which minimizes $E\left[e_{m+2,n}^2\right]$. This property, along with the use of the orthogonality principle, yields:

$$\xi_{m+2} = \xi_m - c_{m+1}E\left[\psi_{m+1,n}\,e_{m,n-1}\right] = \xi_m - 2c_{m+1}^2(1+k_m)\varepsilon_{m+1}. \tag{19}$$

III. NEW REALIZATION

An important consequence of the parameterized solution obtained in Section II is that the desired estimate $\hat{d}_{m,n} = -\gamma'_m Y_{m,n}$ can be expressed in the order recursive form of (17). This expression results in a symmetric lattice/ladder structure involving both forward and backward prediction errors. The form of this solution obtained from (17) corresponds, for $M = 4$, to the linear phase FIR realization shown in Figure 1.

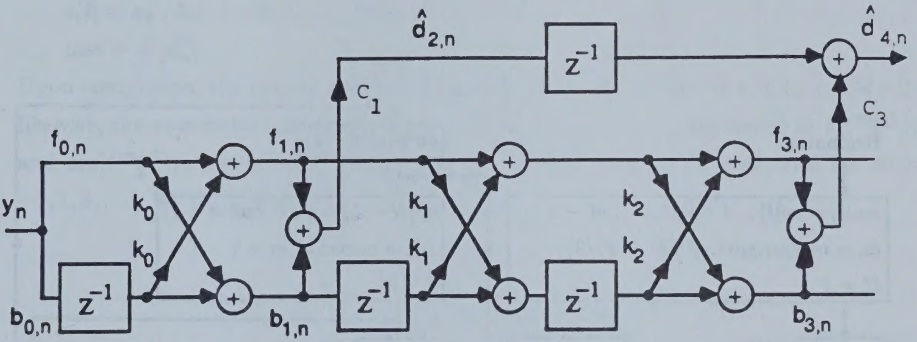


Figure 1: Linear Phase FIR Lattice/Ladder Realization for $M = 4$.

This structure is not canonic in the number of delays. However, it has been shown that the roundoff noise characteristics of a lattice structure are superior to these of a direct form implementation [10]. Therefore, the new structure appears to have attractive numerical properties for linear phase FIR filtering. Both normalized [11] and one-multiplier lattice forms [12] have even better numerical properties [10], [12]. This suggests the use of such techniques along with the procedure of Section II to further enhance the numerical properties. The one-multiplier lattice form can also be used to reduce the required number of multiplications.

IV. IMPLEMENTATION OF FIXED-POINT COMPUTATION

The remarks concerning fixed-point implementation pointed out by Le Roux and Gueguen carry over for the extended formulation (11): the correlation data $\{\rho_i\}$ and $\{\sigma_i\}$ can be scaled so that the variables $\{\zeta_{m,i}\}$, as well as $\{\varepsilon_{m,i}\}$ and $\{k_m\}$, are all bounded by one. This property, however, is not guaranteed for the $\{c_m\}$, which requires careful programming of (15) and appropriate scaling for (17).

To summarize the computation, and for programming purposes, a flowchart of the algorithm presented in Section II is provided in Figure 2. In this flowchart, the result of (10) is stored in either ep[ℓ] or em[ℓ], where:

ep[ℓ] corresponds to $\varepsilon_\ell^+ \triangleq \varepsilon_{m+1,\ell}$, for $\ell = m+2, m+3, \dots, M-1$,

em[ℓ] corresponds to $\varepsilon_\ell^- \triangleq \varepsilon_{m+1,m+1-\ell}$, for $\ell = m+1, m+2, \dots, M-1$.

The result of (15) is stored in z[ℓ], where, $\langle \cdot \rangle$ denoting integer part:

z[ℓ] corresponds to $\zeta'_\ell \triangleq \zeta_{m+2,\ell+\langle\frac{m}{2}\rangle+1}$, for $\ell = \langle\frac{m+3}{2}\rangle, \langle\frac{m+5}{2}\rangle, \dots, \langle\frac{M-1}{2}\rangle$.

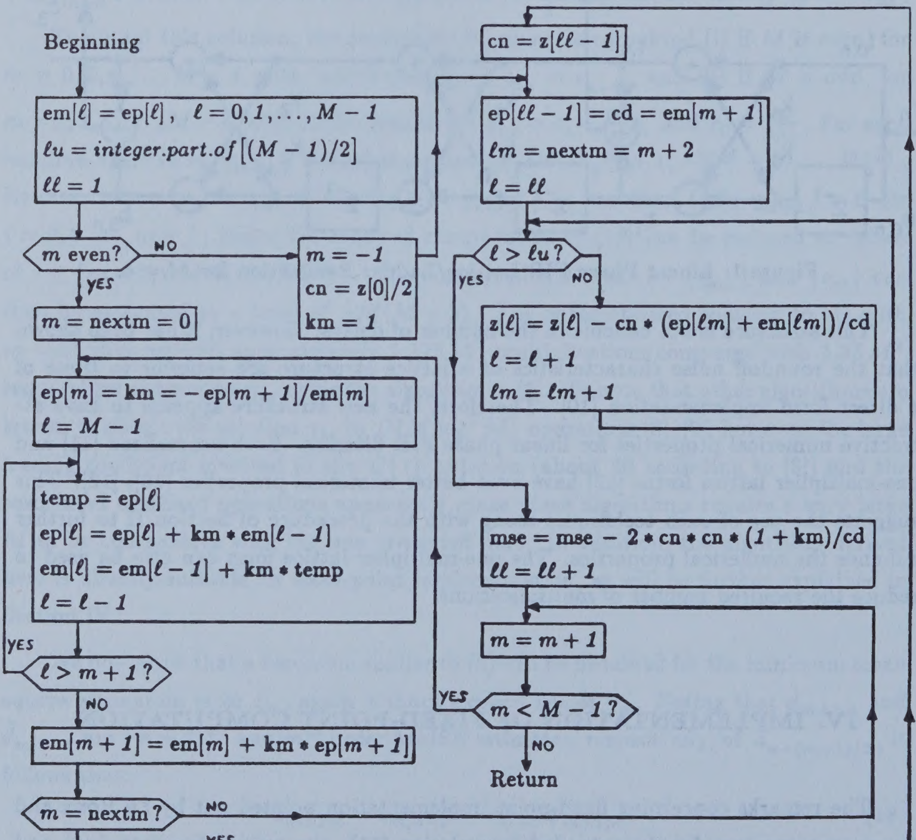

Beginning
em[ℓ] = ep[ℓ], ℓ = 0, 1, …, M − 1
ℓu = integer.part.of[(M − 1)/2]
ℓℓ = 1
m even?
NO
YES
m = −1
cn = z[0]/2
km = 1
m = nextm = 0
ep[m] = km = − ep[m + 1]/em[m]
ℓ = M − 1
temp = ep[ℓ]
ep[ℓ] = ep[ℓ] + km * em[ℓ − 1]
em[ℓ] = em[ℓ − 1] + km * temp
ℓ = ℓ − 1
YES
ℓ > m + 1?
NO
em[m + 1] = em[m] + km * ep[m + 1]
m = nextm?
NO
YES
cn = z[ℓℓ − 1]
ep[ℓℓ − 1] = cd = em[m + 1]
ℓm = nextm = m + 2
ℓ = ℓℓ
ℓ > ℓu?
YES
NO
z[ℓ] = z[ℓ] − cn * (ep[ℓm] + em[ℓm])/cd
ℓ = ℓ + 1
ℓm = ℓm + 1
mse = mse − 2 * cn * cn * (1 + km)/cd
ℓℓ = ℓℓ + 1
m = m + 1
YES
m < M − 1?
NO
Return

Figure 2: Flowchart for Stage-by-Stage Coefficient Computation.

Finally, the results of (8), (16), and (19) are stored in km, cn and cd, and mse, respectively, where:

km corresponds to k_m, and mse to ξ_{m+2},
cn corresponds to $\varsigma_{m,m}$, the numerator of c_{m+1},
cd corresponds to ϵ_{m+1}, the denominator of c_{m+1}.

The arrays ep, em, and z have dimension M, M, and $\langle \frac{M-1}{2} \rangle$, respectively. The initial conditions are:

ep$[\ell] = \rho_\ell$, for $\ell = 0, 1, \ldots, M-1$;
z$[\ell] = \sigma_{\ell+\frac{1}{2}}$, for $\ell = 0, 1, \ldots, \frac{M}{2} - 1$, if M is even;
z$[\ell] = \sigma_\ell$, for $\ell = 0, 1, \ldots, \frac{M-1}{2}$, if M is odd;
mse $= E\left[d_n^2\right]$.

Upon completion, the needed values of k_m are stored in ep$[m]$, for $m = 0, 1, \ldots, M-2$; likewise, the numerator and denominator of each required c_{m+1} are stored in z$[\langle \frac{m+1}{2} \rangle]$ and em$[\langle \frac{m+1}{2} \rangle]$, respectively, for $m = 0, 2, 4, \ldots, M-2$ if M is even, and for $m = -1, 1, 3, \ldots, M-2$ if M is odd.

V. ADAPTIVE FILTERING

An additional feature of the above algorithm is that it provides a linear phase MMSE estimate of order m for $d_{n-(m-1)/2}$ *for each value of* m. Consequently, the design of large order linear phase FIR filters can be achieved by adding additional lattice sections to a lower order design *without* redesigning the previous sections. This stage-by-stage optimization proves extremely useful for adaptive linear phase FIR filtering.

The use of lattice forms is always beneficial in adaptive filtering, due to the nice numerical characteristics of the lattice structure. This accounts for the fact that for non linear phase FIR filtering, structures similar to Figure 1 have been exploited successfully [13], [14] to implement adaptive filters with dramatically improved convergence properties over a direct form adaptive realization. In addition, for MMSE FIR filtering, the stage-by-stage optimality property which results from the design technique allows for algorithms of small dimension. This proves to be an advantage when using stochastic gradient algorithms, which deteriorate in performance when the eigenvalue disparity of the data correlation matrix and/or the dimension of the parameter space increase. For example, for M odd, a direct form linear phase FIR adaptive filter using the Widrow–Hoff LMS (least mean square) algorithm [15] requires one adaptive loop of dimension $(M+1)/2$, with all elements coupled. On the other hand, the structure shown in Figure 1 can be made adaptive using $M + (M+1)/2$ *uncoupled* adaptive loops of one dimension, which appear to have superior convergence properties in many applications.

A suitable stochastic gradient procedure for training k_m to minimize the quantity $E\left[f^2_{m+1,n}\right] + E\left[b^2_{m+1,n}\right]$ is given by:

$$\hat{k}_{m,n+1} = \hat{k}_{m,n} - \mu\left[(f^2_{m,n} + b^2_{m,n-1})\hat{k}_{m,n} + 2f_{m,n}b_{m,n-1}\right], \tag{20}$$

where μ is a positive real constant and $\hat{k}_{m,n}$ is an estimate for k_m at time n. A corresponding algorithm can be used for the coefficients of the FIR filter; thus, a procedure suitable for training c_{m+1} to minimize $E\left[e^2_{m+2,n}\right]$ can be found to be :

$$\hat{c}_{m+1,n+1} = \hat{c}_{m+1,n} - \mu\psi_{m+1,n}(\psi_{m+1,n}\hat{c}_{m+1,n} - e_{m,n-1}). \tag{21}$$

VI. CONCLUSION

We have presented a new parameterization for the solution to the Toeplitz system of linear equations which arises in the MMSE design of a linear phase FIR filter. This parameterization enables one to make natural use of lattice forms in the filter implementation. A fast algorithm has been derived, not only suitable for the fixed-point computation of the solution to the Toeplitz system, but also directly applicable for the MMSE design of a linear phase FIR filter. The computational complexity has been found to compare favorably with existing techniques for solving a set of Toeplitz linear equations. From a more general standpoint, the lattice/ladder FIR realization resulting from the algorithm is appropriate for the implementation of any linear phase FIR filter. Moreover, the new structure has been shown to have improved numerical properties, with respect to a direct form FIR realization.

Adaptive linear phase FIR filters are important in a variety of applications [4], [7]. The realization of Figure 1, because of its symmetric configuration, is ideally suited for their implementation. In this case, we have shown that a *single one-dimensional* adaptive algorithm can be used in Figure 1 to train each parameter *in an uncoupled fashion*, which compares favorably with the stochastic gradient techniques for adaptive linear phase prediction discussed in [4]. In order to minimize the effects of roundoff noise, it is desirable to scale all the data variables to have as large a variance as possible while keeping the probability of an overflow within tolerable limits. It is generally difficult to achieve this goal and to obtain a value for c_{m+1} which is, for example, bounded by one. However, it is possible to implement the quantity $c_{m+1}\psi_{m+1,n}$ by using a fixed-point multiplication followed by a binary shift to obtain the result appropriately scaled. But in general careful programming is required for fixed-point implementation of expressions like (15) and (16).

REFERENCES

[1] A.V. Oppenheim and R.W. Schafer, *Digital Signal Processing*, Englewood Cliffs, NJ: Prentice–Hall, 1975.

[2] D.C. Farden and L.L. Scharf, "Statistical Design of Nonrecursive Digital Filters," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP–22, No. 3, pp. 188–196, June 1974.

[3] D.C. Farden, "The Solution of a Special Set of Hermitian Toeplitz Linear Equations," *ACM Trans. Mathematical Software*, Vol. 3, pp. 159–163, June 1977.

[4] B. Friedlander and M. Morf, "Least Squares Algorithms for Adaptive Linear–Phase Filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP–30, No. 3, pp. 381–390, June 1982.

[5] B. Friedlander, "Lattice Filters for Adaptive Processing," *Proc. IEEE*, Vol. 70, No. 8, pp. 829–867, August 1982.

[6] J. Le Roux and C. Gueguen, "A Fixed Point Computation of Partial Correlation Coefficients," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP–25, No. 3, pp. 257–259, June 1977.

[7] S.L. Marple, Jr., "Fast Algorithms for Linear Prediction and System Identification Filters with Linear Phase," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP–30, No. 6, pp. 942–953, December 1982.

[8] R.R. Bitmead and B.D.O. Anderson, "Asymptotically Fast Solution of Toeplitz and Related Systems of Linear Equations," *Linear Algebra and Its Applications*, Vol. 34, pp. 103–116, 1980.

[9] M. Morf, "Doubling Algorithms for Toeplitz and Related Equations," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Denver, CO, pp. 954–959, April 1980.

[10] J.D. Markel and A.H. Gray, Jr., "Roundoff Noise Characteristics of a Class of Orthogonal Polynomial Structures," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP–23, No. 6, pp. 473–486, October 1975.

[11] A.H. Gray, Jr., and J.D. Markel, "A Normalized Digital Filter Structure," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP–23, No. 3, pp. 268–277, June 1975.

[12] J. Makhoul, "A Class of All–Zero Lattice Digital Filters: Properties and Applications," *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP–26, No. 4, pp. 304–314, August 1978.

[13] L.J. Griffiths, "A Continuously-Adaptive Filter Implemented as a Lattice Structure," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Hartford, CT, pp. 683–686, May 1977.

[14] E.H. Satorius and J.D. Pack, "Application of Least-Squares Lattice Algorithms to Adaptive Equalization," *IEEE Trans. Commun.*, Vol. COM–29, No. 2, pp. 136–142, February 1981.

[15] B. Widrow and M. Hoff, Jr., "Adaptive Switching Circuits," *IRE WESCON Convention Record*, Pt. 4, pp. 96–104, 1960.

SYNTHESIS OF MINIMUM ROUNDOFF

NOISE IN 3-D STATE SPACE DIGITAL FILTERS

A. Zilouchian and R.L. Carroll

George Washington University
Department of Electrical Engineering and Computer Science
Washington, D.C. 20052

Abstract

The optimal synthesis of SISO three dimension (3-D) state space digital filters with fixed-point arithmetic is investigated. Two Lyapunov equations are established for locally controllable, locally observable and stable 3-D filters. The conditions for optimability with respect to the output roundoff noise (with L_2 scaling) are given for these digital filters. It is shown that these conditions have similar forms as the optimal structure design of the 1-D case. The optimal synthesis for SISO separable-in-denominator 3-D filters are also provided.

I. INTRODUCTION

The problem of "optimal realization" of digital filters can be formulated as the search for a class of state space realization which minimizes the output roundoff noise level under a scaling constraint. Different aspects of this realization have been explored by some investigators of 1-D filters (for example [1], [2]). Recently optimal realization of 2-D state-space digital filters have been formulated [3]-[5]. Motivated by the usefulness of 3-D filters in such fields as biomedical, ultrasonic and sonar image processing; in this paper we consider the "optimal synthesis" of some 3-D filters. We use the extension of Roesser's 2-D model [7] as the basis for our 3-D filter in state-space realization since it possesses some unique futures from both the analysis and synthesis point of view.

II. STATE SPACE MODEL AND REALIZATION

Consider the following model as the extension of Roesser's state-space model [7] for a SISO, 3-D discrete filter:

$$\begin{bmatrix} x^h(i+1,j,k) \\ x^v(i,j+1,k) \\ x^d(i,j,k+1) \end{bmatrix} = \begin{bmatrix} A_1 & A_2 & A_3 \\ A_4 & A_5 & A_6 \\ A_7 & A_8 & A_9 \end{bmatrix} \begin{bmatrix} x^h(i,j,k) \\ x^v(i,j,k) \\ x^d(i,j,k) \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} u(i,j,k) \tag{1-a}$$

$$y(i,j,k) = \begin{bmatrix} c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} x^h(i,j,k) \\ x^v(i,j,k) \\ x^d(i,j,k) \end{bmatrix} \qquad (1\text{-b})$$

where $x^h \varepsilon R^{n_1}$, $x^v \varepsilon R^{n_2}$, $x^d \varepsilon R^{n_3}$ are horizontal, vertical and depth state vectors respectively, and i,j,k are non-negative integers. u(i,j,k), y(i,j,k) are the scalar input and output.

We will simplify the notation in a more compact form as follows:

$$x_{11}(i,j,k) = A\, x(i,j,k) + B\, u(i,j,k) \qquad (2\text{-a})$$

$$y(i,j,k) = C\, x(i,j,k) \qquad (2\text{-b})$$

where $x(i,j,k)\ \varepsilon R^q$ with $q = n_1 + n_2 + n_3$ and A,B,C are real matrices of appropriate dimensions. For model (1) the set of boundary conditions consist of $x^v((i,o,o)$, $x^d(i,o,o)$, $x^h(o,j,o)$, $x^d(o,j,o)$, $x^h(o,o,k)$ and $x^v(o,o,k)$ for $i,j,k \geq (0,0,0)$

Definition

The controllability and observability Grammian are defined as:

$$K(i,j,k) = \sum_{\substack{m=o \\ m+n+P \neq 0}}^{i} \sum_{n=o}^{j} \sum_{P=o}^{k} f(m,n,P)\ f^T(m,n,P) \qquad (3)$$

$$W(i,j,k) = \sum_{m=o}^{i} \sum_{n=o}^{j} \sum_{P=o}^{k} g^T(m,n,P)\ g(m,n,P) \qquad (4)$$

where f(m,n,P) is the mapping from the input to the state, and g(m,n,P) is the mapping from weak initial condition (x(o,o,o)) to the output.

Corollary: If a 3-D system is locally controllable (observable), then it's controllability (observability) Grammian are non-singular and symmetric positive definite.

The description of 3-D filters in term of the transfer function are:

$$H(Z_1,Z_2,Z_3) = \frac{y(Z_1,Z_2,Z_3)}{u(Z_1,Z_2,Z_3)} = C^T(S-A)^{-1}B \qquad (5)$$

where $S = Z_1 I_{n_1} \dot{\wedge} Z_2 I_{n_2} \dot{\wedge} Z_3 I_{n_3}$

$\dot{\wedge}$ denotes the direct sum of matrices.

For each transfer function $H(Z_1,\ Z_2,\ Z_3)$, there are infinite realizations. In fact if T is a matrix in the form:

$$T = T^h \overset{\cdot}{\wedge} T^v \overset{\cdot}{\wedge} T^d \tag{6}$$

where T^h, T^v, T^d are $n_1 \times n_1$, $n_2 \times n_2$, and $n_3 \times n_3$ nonsingular matrices, then $(T^{-1}AT, T^{-1}B, CT)$ is a realization from the transformation $x = T^{-1}x$ which doesn't alter the transfer function of the filter.

Any different realization of the transfer function performs in a different way with respect to roundoff accumulation error and also coefficient quantization error under a finite word length constraint. Therefore, it's desirable to select the realization which minimizes the output roundoff error. This realization which is called commonly as "optimal realization" will be considered in the next two sections.

III. SCALING, ROUNDOFF ERROR AND OPTIMAL REALIZATION

In implementation of digital filters with fixed point arithmetic, scaling constraints are required to prevent the overflow at the state variable summing node. This scaling constraint is essential since the dynamic range of the internal register should be bounded by $\pm \varepsilon 2^{m-1}$, where ε is quantization step size and m is the finite register length. Here we use L_2 scaling rule for the 3-D filter. As in the 1-D case [1] with the qth internal register, (whose contents represent the variable x_q), we may associate two impulse response sequences: the first of these sequences $f(i,j,k)$ determines the dynamic range of x_q, while the second sequence $g(i,j,k)$, determines the effects of the output roundoff error introduced into $x_q(i,j,k)$. Furthermore, we also wish to consider the possibility of unequal word length. Let m_q be the number of bits to be alloted to variable $x_q(i,j,k)$ requiring that the average word length be m

$$\sum_{q=1}^{n_1+n_2+n_3} m_q = m(n_1 + n_2 + n_3) \tag{7}$$

$$\delta^2 \|fq\| = \varepsilon\, 2^{m_q - 1} \tag{8}$$

$$\text{with } \|fq\| = \sum_{\substack{i=o}}^{\infty} \sum_{j=o}^{\infty} \sum_{k=o}^{\infty} f_q^2(i,j,k)$$

$$i + j + k \neq 0$$

where δ is the parameter which determined the probability of overflow. Moreover the error variance at the output due to the noise introduced in the state is given as:

$$\sigma_x^2 = \sum_{q=1}^{L} V_q \frac{\varepsilon^2}{12} \|g_q\| \qquad L = n_1 + n_2 + n_3 \tag{9}$$

$$\text{with } \|g_q\| = \sum_{i=o}^{\infty} \sum_{j=o}^{\infty} \sum_{k=o}^{\infty} g_q^2(i,j,k) \tag{10}$$

where $V_q \leq L + 1$ is the number of multiplication needed for computation of x_q.

Theorem 3.1

Let model (1) be stable, locally controllable and locally observable. If the corresponding transfer function is not separable in denominator, then the following Lyapunov equations hold:

$$K = AKA^T + b^* \tag{11}$$

$$W = A^T WA + C^* \tag{12}$$

$$\text{where } b^* = \begin{bmatrix} B_1 B_1^T & & \\ & B_2 B_2^T & \\ & & B_3 B_3^T \end{bmatrix} \tag{13}$$

with

$$K = \lim_{\substack{i \to \infty \\ j \to \infty \\ k \to \infty}} K(i,j,k) \qquad W = \lim_{\substack{i \to \infty \\ j \to \infty \\ k \to \infty}} W(i,j,k) \tag{15}$$

Proof (11) and (12) can be derived using (3), (4) and is given in [6].

K and W are partitioned as

$$K = \begin{bmatrix} K^h & K_2 & K_3 \\ K_2^T & K^v & K_6 \\ K_3^T & K_6^T & K^d \end{bmatrix} \qquad W = \begin{bmatrix} W^h & W_2 & W_3 \\ W_2^T & W^v & W_6 \\ W_3^T & W_6^T & W^d \end{bmatrix} \tag{16}$$

where K^h, W^h, $\epsilon R^{n_1 \times n_1}$ K^v, W^v, $\epsilon R^{n_2 \times n_2}$ K^d, W^d, $\epsilon R^{n_3 \times n_3}$

An examination of (11) and (12) shows that each element of the K matrix is the inner product of sequences f_m and f_n, and each element of W matrix is the inner product of g_m and g_n. Therefore equations (8) and (9) can be written as:

$$\delta^2 K_{qq} = \varepsilon\, 2^{mq^{-1}} \tag{17}$$

$$6_x^2 = \sum_{q=1}^{L} V_q \frac{\varepsilon^2}{12} W_{qq} \tag{18}$$

where $L = n_1 + n_2 + n_3$

Moreover the output roundoff variance of unscaled filters for unequal word length can be derived. Using a diagonal scaling transformation, (18) can be rewritten as:

$$6_x^2 = \frac{L+1}{3}\, \delta^2 \sum_{q=1}^{L} \frac{W_{qq} K_{qq}}{2^{mq}} \tag{19}$$

Synthesis of Optimal Word Length

Applying the arithmetic-geometric mean inequality [1], where the word length is

$$mq = m + \frac{1}{2}\left[\log_2 K_{qq} W_{qq} - \frac{1}{L}\sum_{q=1}^{L} \log_2 K_{q'q'} W_{q'q'}\right] \tag{20}$$

The following alternate expression for (18) is derived:

$$6_x^2 = \frac{(L+1)L}{3}\left(\frac{\delta}{2^m}\right)^2 \left[\prod_{q=1}^{L} K_{qq} W_{qq}\right]^{\frac{1}{L}} =$$

$$M\left[\prod_{i=1}^{n_1} K_{ii}^h W_{ii}^h\right]^{\frac{1}{L}} \left[\prod_{j=1}^{n_2} K_{jj}^v W_{jj}^v\right]^{\frac{1}{L}} \left[\prod_{k=1}^{n_3} K_{kk}^d W_{kk}^d\right]^{\frac{1}{L}} \tag{21}$$

with $M = \dfrac{(L+1)(L)}{3}\left(\dfrac{\delta}{2^m}\right)^2$

Theorem 3.2: 3-D digital filter with optimal word length as (20) have minimum roundoff noise iff K^h, K^v, K^d, W^h, W^v, W^d are in diagonal form.

Proof It is known that for a positive definite symmetric matrix the products of its diagonal is no less than its determinant [1]

$$e(P) = \left[\frac{\det P}{\Pi P_{ii}}\right]^{\frac{1}{2}} \qquad o < e(P) \le 1 \qquad (22)$$

with equality hold iff P is diagonal·(33) can be written as:

$$6_x^2 = M\left[\frac{\det K^hW^h}{[e(k^h)e(w^h)]^2}\cdot\frac{\det K^vW^v}{[e(K^v)e(W^v)]^2}\cdot\frac{\det K^dW^d}{[e(K^d)e(W^d)]^2}\right]^{\frac{1}{L}}$$

$$\ge M \det(K^hW^h)\det(K^vW^v)\det(K^dW^d) \qquad (23)$$

with equality hold iff K^h, W^h, K^v, W^v, K^d, W^d are simultaneously diagonal. Q.E.D.

It is shown that these conditions for an optimal realization can always be met. Since K and W are positive definite, so are K^h, K^v, K^d, W^h, W^v, W^d. Consider a nonsingular transformation in the form

$$T = T^h \dot{\wedge} T^v \dot{\wedge} T^d$$

The change of variable $\hat{x} = T^{-1}x$ doesn't effect the positive real value of

$$\phi = \hat{K}\hat{W} = T^{-1}(KW)T \qquad (24)$$

Moreover T doesn't change the eigenvalues of

$$\phi^h = \hat{K}^h\hat{W}^h \qquad \phi^v = \hat{K}^v\hat{W}^v \qquad \phi^d = \hat{K}^d\hat{W}^d \qquad (25)$$

which are also positive and real. Using proper T^h, T^v and T^d as shown in [1], [8] for 1-D diagonalization process, it's always possible to obtain $\hat{K}$, $\hat{W}$ in the following forms:

$$\hat{K} = T^{-1}KT^{-T} = \begin{bmatrix} I_{n_1} & & X \\ & I_{n_2} & \\ X & & I_{n_3} \end{bmatrix} \qquad (26)$$

$$W' = T^TWT = \left[\begin{array}{c:c:c}
\begin{matrix}\alpha_1 & & 0 \\ & \alpha_2 \ddots & \\ 0 & & \alpha_{n_1}\end{matrix} & & X \\ \hdashline
& \begin{matrix}\beta_1 & & 0 \\ & \beta_2 \ddots & \\ 0 & & \beta_{n_2}\end{matrix} & \\ \hdashline
X & & \begin{matrix}\gamma_1 & & 0 \\ & \gamma_2 \ddots & \\ 0 & & \gamma_{n_3}\end{matrix}
\end{array}\right] \tag{27}$$

X denotes regions of elements which are not necessarily zero. Substitution of (38) and (39) into (35) with equality hold yield:

$$6^2_x = M\left[\prod_{q=1}^{L} S_q\right]^{\frac{1}{L}} \tag{28}$$

where S_q is the diagonal elements of $\hat{W}$

Synthesis of equal wordlength

Using eq. (19) with equal word length yield:

$$6^2_x = \frac{L}{3}\left(\frac{\delta}{2^m}\right)\left[\sum_{q=1}^{L} W_{qq}\,W_{qq}\right]$$

$$= \frac{L}{3}\left(\frac{\delta}{2^m}\right)^2\left[\sum_{q=1}^{n_1} K^h_{qq}W^h_{qq} + \sum_{q=1}^{n_2} K^v_{qq}\,W^v_{qq} + \sum_{q=1}^{n_3} K^d_{qq}\,W^d_{qq}\right] \tag{29}$$

Theorem 3.3 The necessary and sufficient conditions for optimal realization with equal word length of 3-D digital filters are:

$$W^h = \rho_1^2 K^h$$

$$W^v = \rho_2^2\, K^v \tag{30}$$

$$W^d = \rho_3^2\, K^d$$

where ρ_1, ρ_2, ρ_3 are real non-zero constants.

Proof: 6_x^2 is minimized iff each of the summing terms in (29) is minimized. For 1-D filters it was shown that:

$$\left[\frac{1}{n}\sum_{i=1}^{n} K_{ii}W_{ii}\right]^{\frac{1}{2}} \geq M_a \tag{31}$$

$$\text{where } M_a = \frac{1}{n}\sum_{i=1}^{n} \Gamma_i \tag{32}$$

with Γ_i the square roots of eigenvalues of(KW) with equality hold

iff

$$\left\{\begin{array}{ll} K = DWD & \text{for some diagonal matrix D} \\ K_{ii}W_{ii} = K_{jj}W_{jj} & \text{for any i,j} \end{array}\right. \tag{33}$$

Since a sum of terms is minimized by the minimum of each term, eq. (33) applies to each term in (29). The result is (30).

VI. Synthesis of Separable-in-Denominator 3-D Filters

In this section two modified Lyapunov equations, as well as the optimal synthesis of separable-in-denominator filters, are provided.

Definition 4.1 A 3-D filter is called "Completely Separable-in-Denominator" if the denominator or the transfer function is a product of three polynomials, each of which is a function solely of one independent delay.

It is obvious from model (1) and equation (5) that a 3-D System is "completely Separable-in-Denominator" if (A4, A7, A8) or (A2, A3, A6,) be null matrices in model (1).

Definition 4.2 A 3-D system is called "Partially Separable-in-Denominator" if the deonominator of transfer function is the product of two independent polynomials, and are not also completely product separable.

The "partial separability" of (5) can be met if one the pairs (A4, A5), (A7, A8), (A2, A3) or (A5, A6) be equl zero in model (1).

In this section we given two alternative equations for "Complete Separable-in-Denominator" case (A4, A7, A8). The other cases can be treated similarly using definition (3) and (4) the following equations can be derived:

$$K^h = A_1 K^h A_1^T + A_2 K^v A_2^T + A_3 K^d A_3^T + b_1 b_1^T$$

$$K^v = A_5 K^v A_5^T + A_6 K^d A_6^T + b_2 b_2^T \qquad (34)$$

$$K^d = A_9 K^d A_9^T + b_3 b_3^T$$

$$W = A^T [Wd] A + C^T C \qquad (35)$$

where Wd are the block diagonal of W. It is worth noticing that the off diagonal matrices of K are zero while W has non zero off diagonal matrices as well.

Theorem 4.1

A 3-D "complete Separable-in-Denominator" system could be characterized as three 1-D systems as follows:

$$R_1 = (A_1, B_h, C_1)$$

$$R_2 = (A_5, B_v, C_v) \qquad (36)$$

$$R_3 = (A_9, B_3, C_d)$$

where for B_h, B_v, C_v, C_d have been defined properlly and is given in [6].

In the sequel, the optimal synthesis procedures of 1-D can be applied directly using the following theorem.

Theorem 4.2 A 3-d "completely separable-in-denominator" filter with either optimal or equal word length has minimum roundoff noise iff the three 1-D equivalent system R_1, R_2 and R_3 are optimal in 1-D sense.

Proof Similar to section III.

Remark: Since the numbers of multiplications per output usually increase for optimal realizations, for complete "Separable-in-Denominator" always it is possible to use "block optimal structure" to reduce the numbers of multiplications and as the result the level of the output roundoff noise of the digital filters.

Conclusion

The well-known optimal procedure for minimum noise state-space structure of 1-D filters were extended for 3-D fixed-point arithmetic digital filters. The conditions for optimal realization for both optimal and equal word length registers were established. It was shown that there are some differences in the procedure for implementation of these realizations in 3-D digital filters compare to 1-D case. In general the disadvantage of optimal realization is the increasing in the numbers of multiplications especially for high-order filters. To overcome this problem, the optimal block structure form can be implemented in special cases which it is possible to decompose the given 3-D filter as cascade or parallel subfilters.

REFERENCES

[1] C.T. Mullis and R.A. Roberts, "Synthesis of Minimum Roundoff Noise Fixed Point Digital Filters", IEEE Trans. Circuits Syst., Vol. CAS-23, pp. 551-562, Sept. 1976.

[2] L.B. Jackson, A.G. Lindgren and Y. Kim, "Optimal Synthesis of Second-order State Space Structures for Digital Filters", IEEE Trans. Circuits and Systems, Vol. CAS-26, pp. 149-153, Mar. 1979.

[3] B.C. Mertzius, "On the Roundoff Noise in 2-D State Space Digital Filtering", IEEE Trans. Circuits Syst., Vol. CAS-32, pp. 201-204, Feb. 1985.

[4] A. Zilouchian and R.L. Carroll, "On the Optimal Synthesis of 2-D State Space Digital Filters", in IEEE Trans. Circuit and Systems, Nov. 1985.

[5] A. Zilouchian and R.L. Carroll, "On the Roundoff Noise of Separable-in-Denominator 2-D filters", George Washington University, Dept of EE & CS, Report No. IIST-85-13, July 1985.

[6] R.L. Carroll and A. Zilouchian, "Optimal Synthesis of Multidimension State Space Digital Filters: A General Analysis", George Washington University, Dept. of EE & CS, Report No. IIST-85-20, Sept. 1985.

[7] R.P. Roesser, "A Discrete State-space Model for Linear Image Processing", IEEE Trans. Automatic Cont., Vol. 20, pp. 1-10, Feb. 1975.

[8] B.C. Moore, "Principle Component Analysis in Linear Systems Controllability, Observability and Model Reduction", IEEE Trans. Automat. Cont., Vol. AC-26, 17-32, 1981.

QUANTIZATION ERROR REDUCTION IN DIGITAL CONTROL SYSTEMS

Y.A. WU
Hughes Aircraft Company
El Segundo, CA

One of the common problems in a digital control implementation using microprocessors is the finite-wordlength effects that will introduce quantization errors. Quantization errors will be augmented if the D/A converter that is used in the system to convert a digital signal to an analog signal has less bits than the microprocessor.

A technique is invented here to preserve the original wordlength resolution provided by the microprocessor when a D/A converter that has less bits than the microprocessor is used. In this approach, a high frequency dither signal with a constant amplitude is added to the input of the D/A converter. The addition of dither signal can be done either internally via microprocessor or externally via hardware addition.

To illustrate the above proposed idea, a 16-bit microprocessor with a 12-bit D/A converter is assumed in the following. As shown in Figure 1, the microprocessor provides a 16-bit rate signal output at a sampling rate of 60 Hz. The last four least significant bits are truncated since a 12-bit D/A converter is used. Because of this truncation, the quantization error, h, at output of D/A converter is 16 times larger than the original quantization error, q.

To reduce the quantization error, h, or to preserve the original quantization error, q, a high frequency dither signal is added to the input of the 12-bit D/A converter. This can be done by either a software addition or a hardware addition. Shown in Figure 2 is the hardware addition approach. A 12-bit digital adder is used to add two digital numbers: (i) A_0 to A_{11} from the microprocessor outputs; (ii) B_0 from pulsewidth modulated (PWM) dither signal output with B_1 to $B_{11} = 0$. The PWM dither signal is generated as shown in Figure 2 with its duty cycle (τ_d) proportional to the truncated error, e, and period (τ) equal to 1/1500 sec. The truncated error is obtained via a decoder, which receives the four truncated least significant bits from the microprocessor. The output of the 12-bit adder is connected to the input of the 12-bit D/A converter at a clock rate of 1.5 kHz. As a result of this dither signal, the rate command from the D/A converter output carries the original 12-bit signal from the microprocessor plus a PWM signal which contains the truncated 4-bit information. Since the frequency of the dither signal is greater than the stabilization loop bandwidth (30 Hz), the output of the stabilization system will only respond to the average of the PWM signal which is proportional to the truncated error. Hence, at the output of the stabilization system, the rate command signal will have an effective quantization error step equal to q instead of h.

Finally, as mentioned earlier, the above idea can be implemented internally via software addition. This requires the microprocessor to operate at a high clock rate to generate the dither signal and to carry out the software addition. This approach would eliminate the additional hardware at the expense of running at a high clock rate.

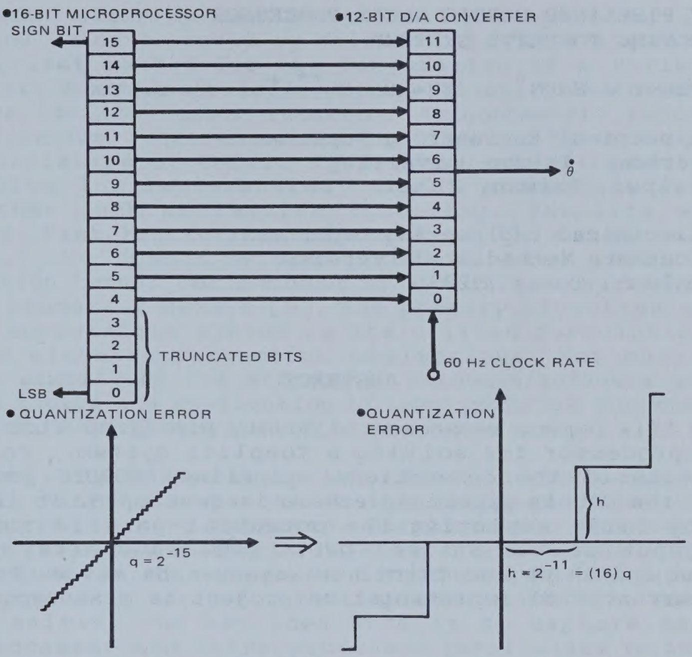

●16-BIT MICROPROCESSOR
SIGN BIT
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
LSB
TRUNCATED BITS
●12-BIT D/A CONVERTER
11
10
9
8
7
6
5
4
3
2
1
0
$\dot{\theta}$
60 Hz CLOCK RATE
● QUANTIZATION ERROR
$q = 2^{-15}$
⇒
●QUANTIZATION ERROR
h
$h = 2^{-11} = (16)\ q$

Figure 1. Quantization errors due to finite-wordlength.

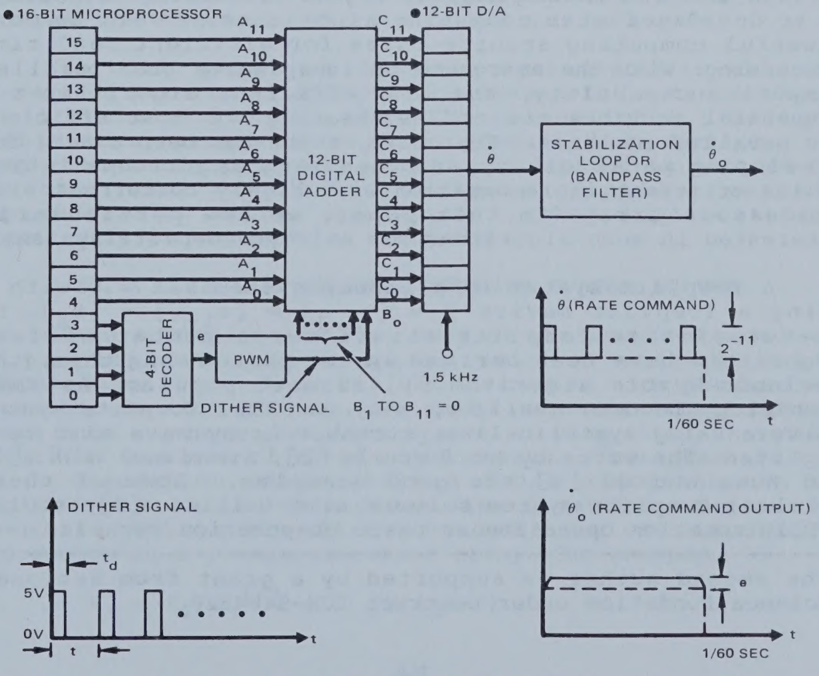

● 16-BIT MICROPROCESSOR
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
A_{11}
A_{10}
A_9
A_8
A_7
A_6
A_5
A_4
A_3
A_2
A_1
A_0
12-BIT DIGITAL ADDER
C_{11}
C_{10}
C_9
C_8
C_7
C_6
C_5
C_4
C_3
C_2
C_1
C_0
●12-BIT D/A
$\dot{\theta}$
STABILIZATION LOOP OR (BANDPASS FILTER)
$\dot{\theta}_o$
4-BIT DECODER
e
PWM
B_o
DITHER SIGNAL
B_1 TO B_{11} = 0
1.5 kHz
$\dot{\theta}$ (RATE COMMAND)
2^{-11}
1/60 SEC
t
DITHER SIGNAL
t_d
5V
0V
t
t
$\dot{\theta}_o$ (RATE COMMAND OUTPUT)
e
t
1/60 SEC

Figure 2. Quantization error reduction — hardware addition approach.

DOUBLY PIPELINED CORDIC ARRAY PROCESSOR FOR SOLVING TOEPLITZ SYSTEMS

TZE-YUN SUNG[*] YU-HEN HU[**,+] H. J. YU[*]

[*]Electrical Engineering Department
National Taiwan University
Taipei, Taiwan, R.O.C.

[**]Electrical Engineering Department
Southern Methodist University
Dallas, Texas 75275

ABSTRACT

In this paper, we present a doubly pipelined VLSI Cordic array processor for solving a Toeplitz system. Follow a discussion of the conventional pipelined CORDIC processor array, the doubly pipelined scheme is developed. It is shown that by fully exploring the potential parallelism, the throughput of the entire CORDIC processor array can be doubled and about one third hardware can be saved. Progress in a current VLSI implementation project is also reported.

I. INTRODUCTION.

The trend of modern digital signal processing technology is to develop faster computing algorithms as well as more powerful computing architectures for efficient real time processing. With the emergence of inexpensive VLSI parallel computing capability, the most efficient algorithms for sequential machines are not necessarily the most efficient for parallel machines. Therefore, there has been a need for developing parallel signal processing algorithms which admits efficient implementation with highly concurrent VLSI processors array. In this paper, we are particularly interested in such algorithms for solving Toeplitz systems.

A Toeplitz system is a linear system $T x = y$ with T being a Toeplitz matirx $T = [t_{i,j}] = [t_{i-j}] = [t_k]$. By virtue of this Toeplitz structure, a number of fast algorithms have been derived in the past. Among them, the Levinson-Durbin algorithm [1] is most popular. As VLSI technology becomes availabe, many parallel Toeplitz system solvers using systolic-like processor array have also been reported. The works by Morf et al. [2], Brent and Luk ,[3] and Kung and Hu [4] are good examples. Some of these parallel Toeplitz system solvers also utilize a hyperbolic CORDIC rotation operation as basic computation kernels.

[+]The second author is supported by a grant from National Science Fundation under contract ECS-8404628.

CORDIC algorithm was first developed by Volder [5] in 1956, and later improved by Walther [6]. This algorithm is an efficient method for the computation of a variety of elementary functions relating to coordinate transformations. Examples include vector rotation, trigonometric functions, multiplication, square roots, and others. These computing characteristics of Cordic iterations make it particular attractive for implementing signal processing related algorithms such as lattice filtering, Toeplitz system solving, Fast Fourier transformation [7], and so on. As a result, the CORDIC algorithm have been taken as a computation kernel for a number of systolic array based VLSI signal processor arrays [8]. The primary advantage of the CORDIC computation system is its unified formulation for various elementary function evaluations. Not only this greatly simplified the efforts in hardware/software design, it also admits the realization of many powerful functions in a regular, highly parallel VLSI processor array.

However, while there are some theoretical works [5],[6], very few physical design examples of the CORDIC processor have been reported. Furthermore, most of these designs follows closely to the conventional CORDIC architecture. In this paper, we introduce a doubly pipelined CORDIC processor array for the implementation of a Toeplitz system solver. The key idea here is to explore maximum interprocessor and intra-processor parallelism to improve the parallel computation efficiency. It will be shown that by doubly pipelining, the computing throughput can be doubled, the pipeline setup latency is significantly reduced, and about one third hardware can be saved.

The basic parallel Toeplitz system solver will be summarized briefly in section II . More details can be found in literatures [2]-[4]. Then the conventional pipelined CORDIC array processor will be discussed in section III. Next, the doubly pipelining scheme is derived in section IV together with some brief analysis. Finally, current implementation status and several other implementation issues will be reported in section V.

II. A GENERIC PIPELINED TOEPLITZ SYSTEM SOLVER

Given a Toeplitz matrix, many of the proposed parallel or pieplined Toeplitz system solvers share similar computation steps. In order not to biased to a particular approach, in this section, we shall summarize these common steps into a generic Toeplitz system solving algorithm. This algorithm consists of three major phases. First phase is the LU decomposition phase where a Schur algorithm can be applied [9]. During this phase, the first two raws of the Toeplitz matrix are first loaded into corresponding processors in a linear processor array. For example,

$$v_i = t_i/t_o^{1/2} \qquad (0 \le i \le N-1) \qquad (1a)$$

$$w_i = t_i/t_o^{1/2} \qquad (0 \leq i \leq N-2) \tag{1b}$$

The next step is to compute the reflection coefficient r(j) of the j-th recursion, or, equivalently, the j-th rotation angles $\Theta(j)$:

$$r(j) = w_{j-1}/v_{j-1} \tag{2a}$$

$$\Theta(j) = \tanh^{-1}(w_{j-1}/v_{j-1}) = \tanh^{-1} r(j) \tag{2b}$$

Then, in parallel or pipelined operation, either of the following computations will be executed in the linear array: That is, for i = j-1, j, ... N-1 do

$$\begin{vmatrix} v'_i \\ w'_i \end{vmatrix} = (1-r^2(j))^{-1/2} \begin{vmatrix} 1 & -r(j) \\ -r(j) & 1 \end{vmatrix} \begin{vmatrix} v_i \\ w_i \end{vmatrix} \tag{3a}$$

$$\begin{vmatrix} v'_i \\ w'_i \end{vmatrix} = \begin{vmatrix} \text{Cosh } \Theta(j) & -\text{Sinh } \Theta(j) \\ -\text{Sinh } \Theta(j) & \text{Cosh } \Theta(j) \end{vmatrix} \begin{vmatrix} v_i \\ w_i \end{vmatrix} = H(\Theta) \begin{vmatrix} v_i \\ w_i \end{vmatrix} \tag{3b}$$

Upon completion, we substitute v_i by v'_i, and w_i by w'_{i+1}. Now, if it is not yet the last (e.g. N-th) iteration, go back to step 2 and repeat above procedure. Otherwise, terminate the LU decomposition phase.

The second and the third phases involve forward and backward substitution procedure for explicit solution of x. Previously, this calls for a linear systolic array with $O(N^2)$ LIFO storage elements [4]. Recently, two novel methods have been suggested. One is proposed by Delosme and Ipsen [10] where an auxiliary vector is used to help computing x. The other method is proposed by Jou and Hu [11] where a lattice filter array is utilized. An implementation based on Delosme and Ipsen's algorithm is shown in figure 1. The details of this figure will be explained later. Please note that in the figure the major processing elements are CORDIC processors which is capable of evaluation the formula in (3b) which is known as "hyperbolic rotation". To see how this operation can be implemented with the help of an array of CORDIC processors, we have the following analysis.

III. PIPELINED CORDIC PROCESSOR FOR TOEPLITZ SYSTEM SOLVER.

To establish connection between CORDIC and rotation based algorithm, let us decompose the angle of rotation Θ into the sum of n sub-angles $\{d_i;\ i = 1, n\}$:

$$\Theta = \sum_{i=0}^{n-1} u_i\, d_i \tag{4}$$

Accordingly, the rotation matrix $H(\Theta)$ can also be decomposed

$$H(\Theta) = \prod_{i=0}^{n-1} H(d_i) \qquad (5)$$

Hence, a single rotation of Θ angle will be replaced by n smaller rotations with d_i angle each. The key idea of CORDIC then is to choose d_i such that (for m = -1)

$$d_i = m^{1/2} \tan^{-1}[m^{1/2} 2^{-s(i)}] = \tanh^{-1} [2^{-s(i)}] \qquad (6)$$

This implies

$$H(d_i) = k_i \begin{vmatrix} 1 & -u_i 2^{-s(i)} \\ -u_i 2^{-s(i)} & 1 \end{vmatrix} \qquad (7)$$

where k_i is a scaling factor. From (7), it is clear that each small rotation can be realized with one simple shift and addition operation. At the end of the n iteration, the desired hyperbolic rotation will be completed. This uniform, powerful rotation mechanism makes the CORDIC ALU extremely attractive for implementing those rotation based algorithms such as the Toeplitz system solver.

In general, the CORDIC algorithm has the following formulation: For i = 0,1,2, ..., n-1, do

$$\begin{vmatrix} x(i+1) \\ y(i+1) \end{vmatrix} = k_i \begin{vmatrix} 1 & m\, u_i 2^{-s(i)} \\ -\, u_i 2^{-s(i)} & 1 \end{vmatrix} \begin{vmatrix} x(i) \\ y(i) \end{vmatrix} \qquad (8)$$

$$z(i+1) = z(i) - u_i\, d_i \qquad (9)$$

where m determines the type of rotations (m=1 for circular, m=-1 for hyperbolic, and m=0 for linear). d_i is chosen as in (6), and and $\{s(i);\ i=0,n-1\}$ is an non-decreasing integer sequence.

Let us now consider how the CORDIC architecture can be fitted into the pipelined Toeplitz system solver. The first step is to compute $\Theta(1) = \tanh^{-1}(w_o/v_o)$. For this, we let $x(0)=v_o$, $y(0)=w_o$, and $z(0)=0$. To facilitate the iteration, u_i is selected such that $|y(i+1)| < |y(i)|$. Hence we choose

$$u_i = \text{sign}\ (x(i)*y(i)) \qquad (10)$$

By induction, at the n-th iteration,

$$z(n) = \tanh^{-1}[w_o/v_o] - \tanh^{-1}[y(n)/x(n)] \qquad (11)$$

For $|y(n)/x(n)| \cong 0$, we have

$$\Theta(j) = z(n) = \tanh^{-1}(w_o/v_o)$$

Next, we would like to compute v'_k and w'_k in (3b). For this, we let $x'(0)=v_k$, $y'(0)=w_k$, $z'(0)=z(n)$. Here, the "'" is used to denote different CORDIC processors. This time, we select u_i such that $|z(i+1)| < |z(i)|$. Thus

$$u'_i = \text{sign}(z(i)) \tag{12}$$

Apply (8) for i=0,1, ..., n, we have

$$\begin{vmatrix} x'(n) \\ \\ y'(n) \end{vmatrix} = K \begin{vmatrix} \text{Cosh } A & -\text{Sinh } A \\ \\ -\text{Sinh } A & \text{Cosh } A \end{vmatrix} \begin{vmatrix} x'(0) \\ \\ y'(0) \end{vmatrix} \tag{13}$$

where $A = z'(0) - z'(n)$ and $K = \prod k_i$. Clearly, if $z'(n) = 0$, then

$$A = z'(0) = z(n) = \tanh^{-1}(w_o/v_o)$$

and

$$x'(n)/K = v_k \text{ Cosh } A - w_k \text{ Sinh } A$$

$$y'(n)/K = w_k \text{ Cosh } A - v_k \text{ Sinh } A$$

will be the desired solution. Based on the above analysis, several observations can be made:

1. The CORDIC algorithm is an approximation algorithm. The accuracy of Θ depends on the condition $|y(n)/x(n)| \sim 0$. So does the accuracy of $x'(n)$ and $y'(n)$ which depends on the condition $|z'(n)| \sim 0$. Hence, it is desirable to make these quantities as close to zero as possible. A detailed error analysis in this aspect can be found in [12].

2. There is a need for normalizaiton due to the presence of the scaling factor K. A simple minded division (with m=0 and n CORDIC iterations) will double the computation time of a single CORDIC operation. However, by taking advantage of the fact that K can be precomputed, the overhead owing to normalization can be minimized. We have investigated this proplem in detail and have some exciting results. Due to page limitations, we refer reader to a report to be published elsewhere [13].

Using CORDIC processor, the Toeplitz system solver can be realized in a straight forward manner: For the j-th iteration, the j-th processor will act as the head processor and compute $\Theta(j)$ in one CORDIC cycle (n CORDIC iterations). Then, $\Theta(j)$ will propagated to the second processor for computing v'_2 and w'_2. Since the second recursion can not be initiated until v'_2 is ready, the head processor will be idle during this period. Thus, a new recursion can be initiated every 2t(C) time units where t(C) is the time duration of one CORDIC cycle. With N-1 recursions, the total computing time will be 2(N-1)t(C). As for hardware requirement, each CORDIC process would need 3 sets of shifter-adders for computing x(i), y(i) and z(i) simultaneously. Thus, about 3N ALUs will be required. In what follows, we shall discuss a more sophisticated doubly pipelined operation which will outperform this classical implementations.

IV. DOUBLY PIPELINED CORDIC PROCESSOR

In this section, we shall demostrate how to modify the CORDIC processor array to accomplish a secondary pipelining of intermediate data. The key idea is to propagate data as soon as they are available for computation [14].

In CORDIC processor implementation, the shift sequence is fixed. Accordingly, the angle rotated in each iteration, via. $\{a_i; i=0,n-1\}$ will also be fixed. We shall further assumed that both the head processor (which accumulates the angle $\theta(j)$) and the other processors in the linear array (which tend to nullify A) using the same set of $\{a_i\}$. Now, let us exam the angle z'(n)

$$z'(n)=z'(0) - \sum_{i=o}^{n-1} u_j a_j = [z(0) - \sum_{i=o}^{n-1} u_j a_j] - \sum_{i=o}^{n-1} u'_j a_j$$

$$= - \sum_{i=o}^{n-1} (u_j + u'_j) a_j$$

It is clear that a simple choice

$$u'_i = -u_i \qquad i=0,1,..,n-1$$

will ensure $z'(n)=0$. Consequently, there will be no need to compute either z'(i) or z(i) during the CORDIC iteration [13]. Instead, the set of $\{u_i; i=1,n-1\}$ can be regarded as an implicit representation of the iteration angle θ.

Using the fact that $\{u_i\}$ will be generated subsequently in the n CORDIC iterations, a "doubly pipelined" CORDIC proocessor array can be devised for the generic Toeplitz system solver. In brief, the operations proceed as follows: When u_0 is decided in the head processor, it will be sent immediately to the second processor. This will prompt the second processor to initiate the computation while the head processor proceeds to compute u_1. By the end of its own first CORDIC iteration, the second processor will relinguish u_1 and send it to processor #3. This, in trun will initiate computation in processor #3. At this moment, processor #2 should continue to work on u_1 sent from the head processor which is in the process of computing u_2. Thus, a pipelining of the u_i is established in the processor array. This will continue until u_{n-1} is computed. This indicates that a $\theta(1)$ is completely specified. Then the processor #2, after one iteration, will complete the first recursion. Next, this processor #2 will become the head processor and proceed to compute $\Theta(2)$. Thus the second recursion is started. This will activate the second computation wavefront of the Teoplitz system solver starting from processor #2. Until the N recursions are compuleted, the desired results will be available in designated storage locations. We call this a doubly pipelined operation because in addition to the pipelining of Toeplitz system sovling computation wavefront [4], the CORDIC iterations are also pipelined.

The doubly pipelined scheme can improve significantly the pipelined computing speed. Since the u_is are propagated to the next neighboring processor every time a CORDIC iteration is completed, the k-th processor in the linear array can be initiated within kt(A) time units. This compares very favorable with the kt(C) = nkt(A) pipeline set up latency required in the earlier pipelined arrays. Moreover, with double pipelining, a new recursion can be initiated every (n+1)t(A) time units. This reduces the total computing time to:

$$(n+1)(N-1)t(A) + nt(A) = (nN+N-1)t(A) = t(C)(N+(N-1)/n)$$

where the second term nt(A) represents the time needed to complete the N-th recursion where only head processor is involved. This above result indicates that the throughput rate is approximately doubled compared with the case without doubly pipelining. A third advantage of doubly pipelining is hardware savings. Since there is no need to evaluate z(i), the third ALU can be removed. Roughly, this is equivalent one-third savings in hardware. Moreover, for propagatin u_i, it needs only one physical wire. But sending a word Θ, at least 8 or 16 parallel wires are needed.

V. IMPLEMENTATION OF TOEPLITZ SYSTEM SOLVER.

Currently, we have an on-going research project for the implemention of a single CORDIC processing element in VLSI. In the data path design, we adopted a dual-ALU architecture which is dicipited in figure 2. By selecting appropriate shift sequence and control flags, this data path chip is capable of performing circular, linear, or hyperbolic rotations computations. A typical operation can be summarized briefly as follows:

To begin with, the S/R switch is activated directing the outside data to be fed into Rxa, Ryb registers (storing x(0)), and Rxb, Rya registers (storing y(0)). This should be performed in the Φ_2 phase. Then, during the following Φ_1 phase, the CORDIC iteration begins. The Barrel shifter and adder in each ALU facilitate the shift and add operation. The result as the output of the adder will be ready by the end of the Φ_1 phase and will be latched into Rx and Ry registers respectively. Then, these results will be routed back to the Rxa, Ryb and Rxb, Rya registers in the subsequent Φ_2 phase. After n (2 phase) clock cycles, x(n) and y(n) will be ready in the Rxa, Ryb and Rxb, Rya registers respectively. The next step then will be the normalization operation.

To accomplish the normalization at the end of rotation, the dual-ALU structure can be de-coupled to operate independently as two ALUs. The pathes swithes SCI thus must be changed accordingly. Due to symmetry, we need to concentrate only on the x portion (left part). For the first clock cycle, Rxa will send its content (x(n)) to overwrite the content in Rxb. In the second clock cycle, then, the content of Rxb will be shifted (scaled). This is achieved by keeping one input of the adder to be zero (grounded). In the following cycles, the content in Rxb will remain constant, while the normalization result accumulating in Rx and Rxa. Using appropriate shift sequence, the normalization cycles can be reduced to 5. Comparing with the number of CORDIC iteration (n = 32 as we used), the cost of normalization is kept minimal. At the end of the normalization, a CORDIC computation is completed.

VI. CONCLUTION.

In this paper, we describe a VLSI Toeplitz system solver using doubly pipelined CORDIC processor array. The architecture is simple and regular, and very suitable for VLSI implementation. Using a doubly pipelining scheme, an improved granularity of pipelined computation is accomplished. It is shown that such a doubly pipelining operation, when compared with the conventional CORDIC pipelining array, the computing speed will be doubled while additional one third of hardware can be saved.

As the progress of this research project, more result in implementation as well as theoretical analysis will be reported in the future publications.

REFERENCE

[1] N. Levinson, "The Wiener RMS (root-mean-square) Error Criterion in Filter Design and Prediction", J. Math. Phys., Vol.25, pp.261-278, Jan. 1974.

[2] J.M. Delosme, M.Morf,"Scattering Array for Matrix Computations", SPIE, Vol.298, Real-Time Signal Processing IV (1981), pp.74-83.

[3] R.P.Brent and F.T.Luk, "A Systolic Array for the Linear-time solution of Toeplitz Systems of equations", Journal of VLSI and Computer Systems, Vol.1, No.1,

pp.1- 22.

[4] S.Y.Kung and Y.H.Hu, " A Highly Concurrent Algorithm and Pipelined Architecture for Solving Toeplitz System", IEEE Trans. on ASSP, Vol.31, No.1, Feb. 1983, pp.66-76.

[5] J.E. Volder, "The CORDIC Trigonomatic Computing Technique", IRE Trans. on Electronic Computers, Vol. EC-8, No.3, Sept. 1959, pp.330-334.

[6] J.S. Walther, "A Unified Algorithm for Elementary Functions", Spring Joint Computer Conference, 1971, pp.379-385.

[7] A.M. Despain, "Fourier Transform Computers using CORDIC Iterations", IEEE Trans. on Computers, Vol. C-23, No.10, pp.993-1001, Oct. 1974.

[8] H.M.Ahmed et al.,"Highly Concurrent Computing Structure for Matrix Arithmatic and Signal Processing", IEEE Computer, Vol.15, No.1, Jan. 1982, pp.65-82.

[9] M. Wax, T. Kailath, "Efficient Inversion of Toeplitz-block Toeplitz Matrix", IEEE Trans. on ASSP Vol. ASSP-31, No.5, pp.1218-1221, Oct. 1983.

[10] J.Delosme et al., "Efficient Parallel Solution of Linear Systems with Hyperbolic Rotations", Research Report, YALEU/DCS/RR-341, Department of Computer Science, Yale University, Nov. 1984.

[11] I-Chang Jou, Yu-Hen Hu, "Lattice Filter Array Implementation of Pipelined Toeplitz System Solver", Tech. Report, Electrical Engineering Department, Southern Methodist University, Dallas, Tx., Sept. 1985.

[12] Yu-Hen Hu, "The Quantization Effects in a VLSI CORDIC Processor", Technical Report, Electrical Engineering Department, Southern Methodist University, Sept. 1985.

[13] Yu-Hen Hu, Tze-Yun Sung and Pin-Kuan Chou, "Implementation of A VLSI CORDIC Processor", Tech. Rpt., Electrical Engineering Dept., Southern Methodist Univ. Sept. 1985.

[14] Yu-Hen Hu, "Pipelined CORDIC Architectures for the Implementation of Rotation-Based Algorithms", Proc. 1985 Symp. on VLSI Tech., Systems and Appl. Taipei, Taiwan, R.O.C. May 1985.

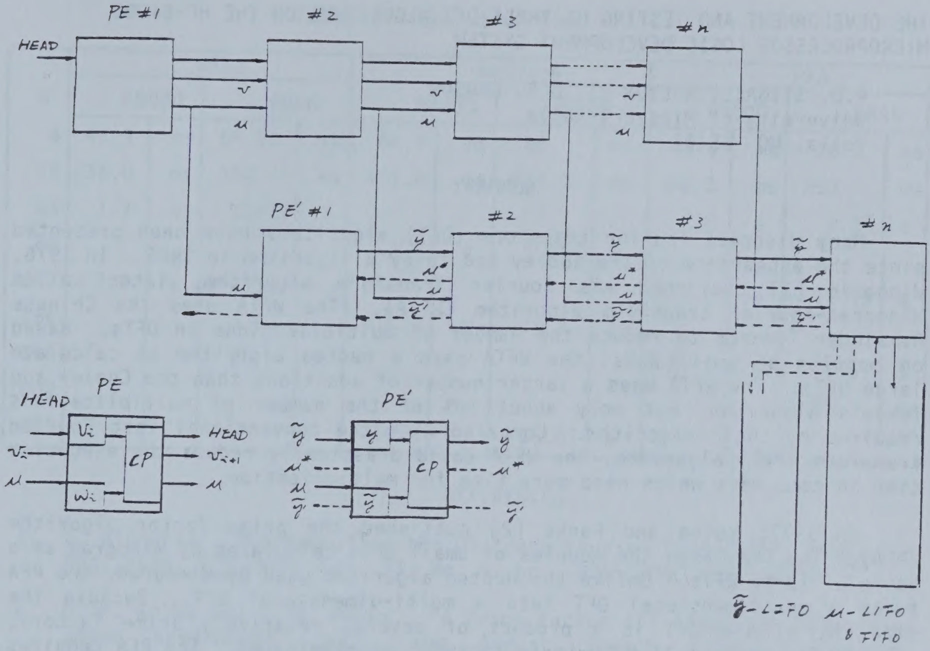

PE #1
#2
#3
#n
HEAD
v
u
PE' #1
#2
#3
#n
ỹ
u*
u
ỹ'
PE
HEAD
vᵢ
u
Vᵢ
Wᵢ
CP
HEAD
vᵢ₊₁
u
PE'
ỹ
u*
u
ỹ'
CP
ỹ-LIFO
u-LIFO & FIFO

Figure 1. Toeplitz System Solver With Doubly Pipelined CORDIC Processor.

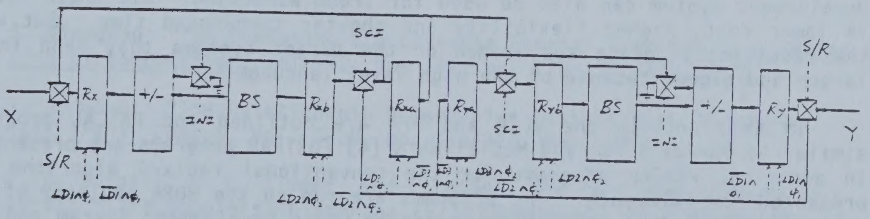

X
S/R
Rx
+/-
INZ
BS
Rxb
SCZ
Rxa
Rya
SCZ
Ryb
BS
INZ
+/-
Ry
S/R
Y
LD1∧φ₁
LD1∧φ₁
LD2∧φ₂
LD2∧φ₂
LD2∧φ₂
LD2∧φ₂
LD1∧φ₁
LD1∧φ₁

Figure 2. Doubly Pipelined CORDIC Processor.

THE DEVELOPMENT AND TESTING OF THREE DFT ALGORITHMS ON THE HP-64000 MICROPROCESSOR LOGIC DEVELOPMENT SYSTEM

P.D. STIGALL, XUEYUN LI, D.F. DAWSON
University of Missouri-Rolla
Rolla, MO 65401

SUMMARY

Many discrete Fourier transform (DFT) algorithms have been presented since the appearance of the Cooley and Tukey's algorithm in 1965. In 1976, Winograd [1] published his Fourier transform algorithm, later called Winograd-Fourier transform algorithm (WFTA). The WFTA uses the Chinese Remainder Theorem to reduce the number of multiplications in DFTs. Based on modules of small DFTs, the WFTA uses a nested algorithm to calculate large DFTs. The WFTA uses a larger number of additions than the Cooley and Tukey's algorithm, but only about 20% of the number of multiplications required by their algorithm. Compared with the conventional fast Fourier transform (FFT) algorithm, the WFTA could drastically reduce the execution time in computers which need more time for multiplication.

In 1977, Kolba and Parks [2] published the prime factor algorithm (PFA). The PFA takes the modules of small DFTs calculated by Winograd as a basis of large DFTs. Unlike the nested algorithm used by Winograd, the PFA turns a one-dimensional DFT into a multi-dimensional DFT. Because the size, N, of the DFT is a product of several relatively prime factors, multiplying by so-called "twiddle factors" is eliminated. The PFA requires less sums of additions and multiplications than does the WFTA so the PFA should perform better in computers which require about the same time for multiplication and addition.

While dedicated FFT chips and boards are available, general purpose microprocessors are a more flexible, convenient and economical way to implement DFTs. Development systems usually support high level languages which make the DFT programming much easier. Many of the facilities of a development system can also be used for troubleshooting. All these result in lower cost, higher flexibility and shorter turnaround time. But when the resulting programs are loaded on the target systems they tend to be larger and slower because of the high level language.

In this article the WFTA and PFA are outlined and PASCAL programs similar to Burrus's [3] and McClellan's [4] FORTRAN programs are presented. In addition, PASCAL programs for the conventional radix-4 algorithm are presented for reference. The programs are run on the 8086 emulator of the Hewlett-Packard (HP) 64000 Microprocessor Logic Development System and the Motorola 68000 Education Computer Board (68000 ECB) after editing, compiling, linking and loading. The results are given in Table 1.

The results demonstrate, of the three algorithms, the PFA is the best for small DFTs while the WFTA is the best for large DFTs as far as the execution time is concerned. The 68000 ECB runs these programs faster than the 8086 emulator.

A DFT system has been produced by using a general purpose microprocessor, a development system and a high level language. Execution times were high because of the software floating point routines and the use of a high level language.

TABLE 1. Execution Time

	WFTA				Radix-4				PFA			
N	68000		8086		68000		8086		68000		8086	
4	42.9	ms	64.25	ms	64.9	ms	86	ms	19.6	ms	28.3	ms
16	235.0	ms	332	ms	461.8	ms	627.5	ms	182.3	ms	253	ms
63	1.7	s	2.375	s					1.8	s	2.55	s
64					2.7	s	3.625	s				
252	7.6	s	10.35	s					8.0	s	11.18	s
256					14.1	s	19.7	s				
840			37	s					33.7	s	46.57	s
1008											55	s
1024					70.2	s	96.67	s				

REFERENCES

1. S. Winograd: "On Computing the Discrete Fourier Transform", Proc. Nat. Acad. Sci. USA, Vol. 73, No. 4, pp. 1005-1006, Apr. 1976.

2. D. P. Kolba and T. W. Parks: "A Prime Factor FFT Algorithm Using High Speed Convolution", IEEE Trans. Acoust. Speech, Signal Processing, Vol. ASSP-25, No. 4, pp. 281-294, Aug. 1981.

3. C. Sidney Burrus, Peter W. Eschenbacher: "An In-Place, In-Order Prime Factor FFT Algorithm", IEEE Trans. Acoust. Speech Signal Processing, Vol. ASSP-29, No. 4, Aug. 1981.

4. J. H. McClellan, H. Nawab: "Complex General-N Winograd Fourier Transform Algorithm (WFTA)", Programs for Digital Signal Processing, New York: IEEE Press, 1979.

5. S. Winograd: "On Computing the Discrete Fourier Transform", Mathematics of Computation, Vol. 32, No. 141, pp. 175-199, Jan. 1978.

6. I. J. Good: "The Relationship Between Two Fast Fourier Transforms", IEEE Trans. Comput. Vol. C-20, pp. 310-317, Mar. 1971.

7. P. D. Stigall, R. E. Ziemer, and L. Hudec: "A Performance Study of 16-Bit Microcomputer-Implemented FFT Algorithms", IEEE MICRO, Vol. 2, No. 1, pp. 61-66, Nov. 1982.

BILINEAR PROGRAMMING AND STRUCTURED STOCHASTIC GAMES

JERZY A. FILAR and TODD A. SCHULTZ
Department of Mathematical Sciences
The Johns Hopkins University
Baltimore, Maryland

Abstract. "One step" algorithms are presented for two classes of structured stochastic games, namely those with additive rewards and transitions, and those which have switching controllers. Solutions to such classes of games under the "average reward" criterion can be derived from optimal solutions to appropriate bilinear programs. The validity of using bilinear programming as a solution method follows from two preliminary theorems, the first of which is a complete classification of undiscounted stochastic games with optimal stationary strategies. The second of these preliminary theorems relaxes the conditions of the classification theorem for certain classes of stochastic games and provides the basis for the bilinear programming results. Analogous results hold for the discounted stochastic games with the above special structures.

This work was supported in part by the Air Force Office of Scientific Research and the National Science Foundation under the grant #ECS-8503440. The paper will appear in the Journal of Optimization Theory and Applications.

PIECEWISE LINEAR MODELING OF MULTIDIMENSIONAL STOCHASTIC NONLINEAR SYSTEMS

A. H. HADDAD AND E. I. VERRIEST
Georgia Institute of Technology
Atlanta, GA 30332-0250

ABSTRACT

This paper extends to the multi-dimensional case an approximation of nonlinear dynamic systems with random inputs, by a set of linear models with Markovian transitions among the models.

SUMMARY

The problem of piece-wise linear approximation of nonlinear dynamic systems subject to stochastic input has been considered for the scalar case in [1-2]. The approach was used to obtain an approximate model for the discrete-time nonlinear system described by Markovian transitions among linear models. The resulting approximation was then utilized to derive an approximate filtering scheme for the nonlinear system. The analysis has been restricted to the scalar case for simplicity, and to illustrate the concepts. However, the extension to the multi-dimensional case may appear to be simple in principle, but it presents several difficulties. The objective of this paper is to provide a systematic approach to the approximate multivariable modeling problem for nonlinear systems driven by stochastic inputs. The paper also briefly describes the applications of the model to the approximate solution of the nonlinear filtering problem.

The discrete-time nonlinear system considered has the form

$$x_{k+1} = f(x_k) + w_k \qquad (1)$$

where x_k is the n-dimensional state vector, and $\{w_k\}$ is a white Gaussian noise sequence with zero mean and covariance matrix Q. The nonlinearity f(x) is assumed to be given by a piece-wise linear approximation

$$f(x) = A_i\, x + b_i, \qquad x \in \Omega_i. \qquad (2)$$

The properties and nature of the approximating model for the scalar case depend on the stability or instability of the linear models, and the size of their regions, Ω_i, relative to the noise standard deviation. In order to extend the notion of stability and size of the regions to the multi-dimensional case a new definition as to stability is needed, and a more systematic approach to the region definition is required.

First a measure $\mu(\cdot)$ on R^n is defined relative to noise covariance matrix Q. If the covariance matrix is singular the measure may be based on the covariance of the state of an equivalent linear system that is obtained from (1) by least squares approximations. The regions Ω_i may now be classified as to their being contracting or expanding (as opposed to stable or unstable), using modified assumptions. A region Ω_i is said to be Contracting (C) if it satisfies the assumption:

(C1) $\mu[f(\Omega_i)] \leq \mu(\Omega_i)$

Similarly, a region Ω_i is said to be Expanding (E) if it satisfies

(E1) $\mu[f(\Omega_i)] > \mu(\Omega_i)$

where the the notation $f(\Omega_i)$ is used for the image of the region Ω_i under the transformation f(x). The class of nonlinearities considered here is further restricted to satisfy:

(C2) $f(\Omega_i) \subset \Omega_i$ (C3) $\mu(\Omega_i) \gg 1$
(E2) $\Omega_i \subset f(\Omega_i)$ (E3) $\mu(\Omega_i) \ll 1$

for contracting and expanding regions, respectively. These assumptions (similar to the scalar case) restrict the class of nonlinearities, but are required for the validity of the approximate Markov model.

The next step is then to derive the Markov transition probabilities among the linear models given in (2), after the restriction of (2) to the regions Ω_i is removed. The model parameters are obtained by micro-partitioning the space R^n into cells whose measure is commensurate with unity relative to the covariance Q. The transition probabilities for a contracting region can then be approximated by considering the number of cells in an expanding region that forms the closure of the mapping of the original contracting region under f. Similarly, the transition probabilities for an expanding region can be obtained via the number of cells that are contained in the contracting region forming the mapping of the expanding region under the transformation f when the noise effects are included.

The resulting model is similar to the scalar case except that the transitions are no longer restricted to neighboring regions only as discussed in [2]. The assumptions (C) and (E) ensure that the transition matrix is relatively sparse. One problem which needs to be further resolved is the case when the matrix A_i is contracting in one direction but expanding in another. Preliminary investigation of this aspect is being considered.

The next phase involves the application of the model to nonlinear filtering problems. Here again, a departure from the scalar case occurs due to the dependence of the transition probabilities on the value of x. The state of the system is determined by x, its covariance P under the appropriate linear model, and the probabilities p_i of the ith linear model. The time update in the filtering problem solution is based on updating P corresponding to the model in effect for the given state, and then updating the probability vector p based on the ellipsoid defined by P, the state estimate, and the allowable regions for transitions. The overall estimate update is obtained as a weighted sum of the allowable transition regions resulting from the current region and P. The measurement update follows similarly by updating the estimate based on the possible models, while the update of the probability vector p is more ad hoc. The update of p is based on a weighted sum of the steady state probability obtained from the original Markov model, and the possible transitions determined from the estimate of x and its covariance under the ith model. Additional study of the resulting filter and its properties is needed.

There are two additional major problems to be considered. The first is involved in relaxing some of the restrictions that limit the types of the nonlinearities which can be used. The second is involved with the identification problem when the nonlinearity is unknown. In both cases, the dimensionality problem can be resolved as in the scalar case utilizing the sparseness of the transition matrix.

ACKNOWLEDGEMENT

This research is supported by the US Air Force Armament Laboratory, under Contract F08635-84-C-0273.

REFERENCES

[1] A. H. Haddad and E. I. Verriest, "Linear Markov Approximations for Piecewise Linear Stochastic Systems," Proc. Ann. Conf. on Inform. Sci. and Systems, pp. 202-206, March 1984, Princeton University.

[2] A. H. Haddad, "On the Modeling and Filtering for Piecewise Linear Stochastic Systems," Proc. 19th Ann. Conf. on Inform. Sci. and Systems, pp. 44-49, March 1985, Johns Hopkins University.

SOME NEW RESULTS IN OVERLAPPING PARAMETRIZATION OF MULTIVARIABLE SYSTEMS

S. Bingulac
Department of Electrical Engineering
Virginia Tech
Blacksburg, Virginia 24061

R. Krtolica
Mathematical Institute, Academy of Science
K. Mihailova 35, Belgrade, Yugoslavia

SUMMARY

Parametrization of State-space and input-output finite dimensional models is one of the main problems in off- and on-line identification of linear, time-invariant MIMO systems. Convenient parametrizations should provide a uniquely identifiable system representation with a small number of parameters. For MIMO systems represented by discrete model:

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t) \quad ; \; x(t) \varepsilon R^n \; ; \; y(t) \varepsilon R^p \; ; \; u(t) \varepsilon R^m \\ y(t) &= Cx(t) \end{aligned} \tag{1}$$

different parametrizations are derived by selecting the independent rows of the Hankel matrix $H_{\infty,\infty}$ or the observability matrix $Q(A,C)$, where:

$$H_{\infty,\infty} = \begin{vmatrix} H_1 & H_2 & \cdots \\ H_2 & H_3 & \cdots \\ \cdot & \cdot & \cdots \end{vmatrix} \; ; \; Q_o(A,C) = \left| C^T \,\vdots\, C^T \; A^T \,\vdots\, \cdots \,\vdots\, C^T \; (A^T)^{n-1} \right|^T \tag{2}$$

$$H_i = C\,A^{i-1}\,B$$

These row selections are specified by a set of integers μ_i, $i=1,2, .., p$, $\mu_i \geq 1$; $\mu_1 + \mu_2 + \ldots + \mu_p = n$, called structure indices. Parametrizations in which structural indices become Kronecker invariants (observability indices) lead to canonical (nonoverlapping) system representations. Identification of canonical forms may cause numerical problems, particularly on the boundaries between two parametrizations, due to the fact that no system has more than one canonical representation. This fact has brought attention to the overlapping parametrizations leading to pseudocanonical forms /1/ /2/ where several sets $\underline{\mu}$, $\underline{\mu} := \{\mu_1, .., \mu_p\}$, of structure indices may represent one particular system. In /3/ for a given set $\underline{\mu}$ it has been shown that for MIMO system (1) the following expression relating available input-output samples $\{u(t),y(t)\}$, $t = 1,2, \ldots, T$, holds:

$$\begin{vmatrix} y_1(t+\mu_1) \\ \vdots \\ y_p(t+\mu_p) \end{vmatrix} = \left| B^* \;\vdots\; A^* \right| z(t) \tag{3}$$

In (3) the $(m\,\mu_p + n)$-dimensional column $z(t)$ is defined by:

$$z^T(t) = \left| u^T(t) \,\vdots\, .. \,\vdots\, u^T(t+\mu_p-1) \,\vdots\, \tilde{y}^T(t) \,\vdots\, .. \,\vdots\, \tilde{y}^T(t+j-1) \,\vdots\, .. \,\vdots\, \tilde{y}^T(t+\mu_p-1) \right|^T ,$$

where $\tilde{y}(t+j-1)$, $j=1, .., \mu_p$, is the p_j-dimensional column, $1 \leq p_j \leq p$, containing only those components $y_i(t+j-1)$ of the output vector $y(t+j-1)$ whose indices i satisfy: $\mu_i \geq j$. For notational convenience and without loss of generality it has been assumed in (3) that: $\mu_i \leq u_{i+1}$, $i=1, .., p-1$.

Similarly as the selection of rows from the $H_{\infty,\infty}$, the parameters in matrices B* and A*, eq(3), uniquely determine the system (1). Therefore, to obtain a pseudocanonical form it is just necessary to find a set $\underline{\mu}$ of structure indices such that the corresponding rows of $H_{\infty,\infty}$ or $Q_0(A,C)$ are linearly independent. The obtained parametrization will represent all systems of given order in a large neighborhood of nominal values of system parameters.

Several practical rules for selecting structure indices have been proposed /1/, but still considerable freedom of choice has been left. Therefore, tedious trail-and-error methods are used in parameter identification to choose structure indices that correspond to a numerically correct base. It is evident that an appropriate description of a class of structure indices leading to linearly independent rows of $H_{\infty,\infty}$ or $Q_0(A,C)$ may significantly save numerical efforts. Having this intention in mind, this paper presents the characterization of the class of structure indices leading to selection of independent rows. This class of structure indices is referred to as the admissible pseudoobservability indices, since they lead to pseudocanonical forms. The characterization of admissible pseudoobservability indices is in this paper done by formulating a necessary condition that any set of indices $\underline{\mu}$ should satisfy. To this end the individual observability index v_i, $i=1,2, .., p$, is defined for each output $y_i(t)$.

It is shown that the individual observability indices have simple interpretation when the dynamic equations are in the Jordan form, $\{A_j, B_j, C_j\} = \{T^{-1}AT, T^{-1}B, CT\}$, T being modal (eigenvector) matrix. According to the formulated necessary condition, the admissible pseudoobservability index μ_i corresponding to $y_i(t)$ should not be greater than the individual observability index v_i for this output, i.e. $\mu_i \leq v_i$.

The individual observability indices v_i, $i=1, .., p$, may be easily evaluated by inspecting the output matrix C_j in the Jordan form realization or by estimating the orders of minimal realizations of MISO systems obtained from a given MIMO system by replacing the output vector $y(t)$ with its single component $y_i(t)$.

REFERENCES

[1] M. Gevers, V. Wertz, "Uniquely identibiable State-space and ARMA parametrizations for Multivariable linear systems", AUTOMATICA, Vol. 20, No. 3, pp. 333-347, 1984.

[2] K. Glover, J. C. Willems, "Parametrization of linear dynamical systems, canonical forms and identifiability", IEEE Trans. Auto. Control, Vol. AC-19, No. 6, pp. 640-645, 1974.

[3] S. Bingulac, R. Krtolica, "Generalized ARMA model for MIMO system identification", Proc. ACC, Boston, Paper No. FA6-10:15, 1985.

STABILITY PROPERTY OF THE CERTAINTY EQUIVALENT CONTROL OF DIGITAL RANDOM-PARAMETER SYSTEMS

ENGIN YAZ
Electrical Engineering Department, University of Arkansas, Fayetteville
AR 72701

Conditions are given under which linear discrete-time systems disturbed by scalar multiplicative noises, can be stochastically stabilized by finite and infinite horizon certainty-equivalent controllers. Stabilizability by a certainty equivalence controller implies stabilizability by the corresponding optimal controller.

SUMMARY

The following random-parameter linear system

$$x_{k+1}=a_kAx_k+b_kBu_k \tag{1}$$

is examined from the viewpoint of stabilizability by certainty equivalent controllers. In(1), $x_k \varepsilon R^n$ and $u_k \varepsilon R^m$ are state and control vectors, respectively. A and B are constant matrices, a_k and b_k are scalar white noise sequences which are mutually uncorrelated. The noise sequences have stationary characteristics, their expected values being $E(a_o)\neq 0$ and $E(b_o)\neq 0$ and variances Σ_a and Σ_b being finite.

The possibility of stabilization of (1) by infinite-horizon controllers was investigated in [1] and [2]. The use of constant feedback, finite sliding-horizon optimal controllers was introduced in [3]. In the present work, we are interested in the certainty equivalent control of system (1), by which we mean that the controller which is constructed based on the system constraint

$$z_{k+1}=E(a_o)Az_k+E(b_o)Bu_k \tag{2}$$

in which the random quantities of (1) are replaced by their expected values. So, the assumption of nonzero means prevents a trivial problem formulation.

When the following infinite-horizon criterion

$$J=\sum_{i=o}^{\infty} z_i^TQz_i+u_i^TRu_i \tag{3}$$

for $Q^T=Q>0$, and $R^T= R>0$ is minimized with respect to system (2), the certainty equivalence controller which will be used in(1) is obtained as

$$u_k=-(R+E^2(b_o)B^TPB)^{-1}E(a_o)E(b_o)B^TPAx_k=Kx_k \tag{4}$$

where P is found from

$$P=E^2(a_o)A^TPA+Q-E^2(a_o)E^2(b_o)A^TPB(E^2(b_o)B^TPB+R)^{-1}B^TPA \tag{5}$$

If we let $(E(a_o)A,E(b_o)B)$be stabilizable and since $Q>0$, (5) has a unique solution $P^T=P>0$, and the following theorem can be proved:

Theorem 1. Under the above assumptions, the stochastic system (1) is stabilizable both in almost sure and mean square asymptotic senses by (4) if both

$$\Sigma_a A^T P A < Q, \text{ and } \Sigma_b B^T P B < R \tag{6}$$

hold.

The next theorem says that the optimal controller is as good as a certainty equivalent controller in stabilizing (1):

Theorem 2. If system (1) is stabilizable by (4) then it is also stabilizable by the optimal control which is obtained by minimizing E(J) with respect to (1). Next, the finite sliding horizon criterion

$$J_R = \sum_{i=k}^{k+N-1} z_i^T Q z_i + u_i^T R u_i + z_{N+k}^T F z_{N+k} \tag{7}$$

with $Q^T = Q > 0$, $R^T = R > 0$, and $F^T = F > 0$, is minimized. The system constraint is given by (2).

The certainty equivalence control is

$$u_k = -(R + E^2(b_o)\, B^T P_{N-1} B)^{-1} E(a_o)\, E(b_o)\, B^T P_{N-1} A x_k = K_R x_k \tag{8}$$

where P_{N-1} is the solution $(N \geqslant 1)$ of

$$P_{i+1} = E^2(a_o) A^T P_i A + Q - E^2(a_o) E^2(b_o) A^T P_i B (E^2(b_o)\, B^T P_i B + R)^{-1} B^T P_i A \tag{9}$$

with $P_o = F$. The optimal solution to this problem is obtained by minimizing $E(J_R)$ with (1) as the system constraint and was treated in [3]. In the finite-horizon case, the following is true:

Theorem 3. Let $P_o \geqslant P_1$. If (6) holds with P replaced by P_{N-1}, then (1) is stabilizable by (8) in both senses of almost sure and mean square asymptotic stability for any $N \geqslant 1$. Similar to the infinite-horizon case, stabilizability by the certainty equivalence controller implies stabilizability by the optimal controller.

Theorems 1 and 3 are still valid if there is an additive white noise vector, but then stability is no longer asymptotic. Also for the infinite-horizon case, the cost criterion must be changed to a long-term average cost criterion. Further extension of these results to be correlated noise sequences is possible by incorporating cross-terms in the performance criteria.

REFERENCES

1. R.T. Ku and M. Athans, IEEE Trans. Autom. Contr., AC-22, pp. 866-868, 1977.
2. W.L. DeKoning, Automatica, 18, pp. 443-453, 1982.
3. E. Yaz, Systems and Contr. Lett.s, 5, pp. 321-326, 1985.

APPLICATIONS OF SINGULAR VALUE DECOMPOSITION TO THE APPROXIMATE PARTIAL MINIMAL REALIZATION PROBLEM

G.M. PITSTICK, J.R. CRUZ and R.J. MULHOLLAND
School of Electrical Engineering and Computer Science
The University of Oklahoma
Norman, OK 73019

ABSTRACT

A review of presently available singular value decomposition (SVD) based approximate realization algorithms is presented, including a new version of Kung's algorithm. Numerical examples are given which illustrate their relative performance in terms of the squared error of the impulse response.

INTRODUCTION

The approximate partial minimal realization (APMR) problem can be stated as follows: given a finite sequence of q x p matrices contaminated by noise $(\underline{\tilde{S}}i)_{i=1}^{2n}$, determine a system with triple $[\underline{A}, \underline{B}, \underline{C}]$ of minimal order that has the impulse response

$$\underline{h}(k) = \underline{S}_k\ \underline{u}(0) \qquad k = 1,2,\ldots \tag{1}$$

The APMR is a classical problem for which there is no completely satisfactory solution. Several algorithms are currently available which provide minimal realizations when the impulse response data is noise free. However, the recursive algorithms are numerically unstable and none of them are suited for the approximate realization problem. Recently, de Jong presented a recursive numerically stable algorithm which can be extended to noisy data. Unfortunately such extension involves an optimization problem with a large number of variables making it impractical for most signal processing applications.

The common link between all these algorithms is that they are based on some decomposition of a Hankel matrix. Among all the decomposition algorithms, the singular value decomposition (SVD) is the most stable and particularly well suited to handle round-off errors and noisy data. It seems therefore natural, to consider the SVD approach to the approximate realization problem.

SVD BASED ALGORITHMS

The use of SVD in this context was first suggested by Zeiger and McEwen [1] in a short note on Ho's algorithm. More recently Kung [2] proposed another algorithm which has been used by Mendel [3] in a modified form. A new version of this algorithm has been suggested [4] which exhibits the same accuracy of the original algorithm and gives stable models without using the assumption of Mendel's version.

For the single input, single output case all these algorithms begin by constructing a Hankel matrix $\underline{H}$ from the noisy Markov parameters and then determine its singular value decomposition, i.e.,

$$\underline{H} = \underline{U}\ \underline{S}\ \underline{V}^T \tag{2}$$

where $\underline{U}$ and $\underline{V}$ are orthogonal matrices and $\underline{S}$ is a diagonal matrix of the singular values of $\underline{H}$. The n x n matrix of rank r that is closest to $\underline{H}$ in the Frobenius norm is

$$\underline{H}' = \underline{U}_1\ \underline{S}_1\ \underline{V}_1^T \tag{3}$$

where $\underline{V}_1$ is the first r columns of $\underline{U}$, $\underline{S}_1$ is the r x r diagonal submatrix of $\underline{S}$ of the r largest singular values of $\underline{H}$ and $\underline{V}_1$ is the first r columns of $\underline{V}$. The observability and controllability matrices are then approximated by $\underline{G} = \underline{U}_1 \underline{S}_1^{\frac{1}{2}}$ and $\underline{E} = \underline{S}_1^{\frac{1}{2}} \underline{V}_1^T$ respectively. Then, the first row of the observability matrix can be used as an estimate for $\underline{c}$ and the first column of the controllability matrix can be used as an estimate for $\underline{b}$.

The differences between the algorithms are confined to the computation of the system matrix $\underline{A}$. Zeiger and McEwen obtain an estimate for $\underline{A}$ by solving the equation

$$\underline{G}\,\underline{A}\,\underline{E} = \underline{H}_1 \tag{4}$$

where $\underline{H}_1$ is $\underline{H}$ shifted one column to the left. Kung's estimate for $\underline{A}$ is obtained by solving the equation

$$\underline{G}_f\,\underline{A} = \underline{G}_\ell \tag{5}$$

where $\underline{G}_f$ consists of the first n-1 rows of $\underline{G}$ and $\underline{G}_\ell$ of the last n-1 rows of $\underline{G}$. Mendel's version of the algorithm solves the equation

$$\underline{G}\,\underline{A} = \underline{G}^{\uparrow} \tag{6}$$

where $\underline{G}^{\uparrow}$ is $\underline{G}$ shifted one row up with an added zero bottom row. The algorithm proposed in [4] uses equation (5) with a modified Hankel matrix $\underline{H}$ which has the *convergence-to-zero* property built into it.

NUMERICAL EXAMPLE

For illustration purposes the four algorithms discussed above were applied to the seismic source wavelet data used by Kung [2]. The squared error of the impulse response was computed for the four realizations giving 74.67×10^{-3} for Zeiger and McEwen's, 1.64×10^{-3} for Kung's, 2.56×10^{-3} for Mendel's and 1.70×10^{-3} for the algorithm proposed in [4].

ACKNOWLEDGEMENT

This work was supported in part by a grant from the Energy Resources Institute and a Summer Research Fellowship from The University of Oklahoma.

REFERENCES

[1] H.P. Zeiger and A.J. McEwen, "Approximate linear realizations of given dimension via Ho's algorithm," **IEEE Trans. Automatic Control**, vol. AC-19, p.153, 1974.

[2] S. Kung, "A new identification and model reduction algorithm via singular value decompositions," in **Proc. 12th Asilomar Conf. Circuits, Systems and Computers**, Pacific Grove, CA, pp. 705-714, 1978.

[3] J.M. Mendel, "Minimum-variance and maximum likelihood recursive waveshaping," **IEEE Trans. Acoust., Speech, Signal Processing**, vol. ASSP-31, pp. 599-604, 1983.

[4] G.M. Pitstick, J.R. Cruz and R.J. Mulholland, "Approximate realization algorithms for truncated impulse response data," submitted to **IEEE Trans. Acoust., Speech, Signal Processing.**

A NEW METHOD FOR ROBUST STABILITY

QIU HUANG & R. LIU
University of Notre Dame
Notre Dame, IN 46556

SUMMARY

Let G be a ring with identity (ring of linear systems). Let H be a subring of G (set of stable systems). Consider the feedback system S(p,c) in Figure 1. The stabilization problem is to find $c \in G$ for a given $p \in G$ such that

$$he_i u_j \in H \quad , \quad i,j = 1,2 \tag{1}$$

u.t.c., S(p,c) is said to be H-stable.

Let p_a be the actual plant, whose model may or may not be known. Let p_o be a model of the plant which is to be used for the stability design. The main question is whether or not the H-stability of $S(p_o,C)$ implies the H-stability for $S(p_a,c)$.

Let

$$\Delta p = p_a - p_o \tag{2}$$

be the deviation between the actual plant and its model. The following Corollary follows from Theorem 1 of [1].

Corollary 1. Let $p_o \in H$, $c \in G$ and $S(p_o,C)$ be H-stable. Then $S(p_a,C)$ is H-stable if and only if

$$M = \Delta p[1 + Q\Delta p]^{-1} \in H \tag{3}$$

where

$$Q = C(1 + p_oC)^{-1} \in H \tag{4}$$

which depends only on the model p_o.

Remarks:

1. Eq.(3) has feedback diagram interpretation given in Figure 2.

2. Desoer and Liu have given a sufficient condition in [2], as well as for the nonlinear case. Bhaya and Desoer [3] considered a stronger H^*-stability condition. Under the stronger stability condition, they also arrived the same n.a.s. condition.

Corollary 1 has important applications to the design of FBS. For example, if p_a is unstable and its coprime fractional representation can not be obtained, such as in the time-varying and/or multidimensional systems. Then Corollary 1 can be used as an alternate method for the design. Given $p_a \in G$, we can choose $p_o \in H$ such that $\Delta p = p_a - p_o$ can be stabilized by some $Q \in H$ as shown in Figure 2. Then

$$C = Q(1 - P_oQ)^{-1} \tag{5}$$

will stabilize $S(p_a,C)$ as well as $S(p_o,C)$. This has an important application to the multi-variate case.

Let p_a be a multivariate transfer function matrix, and let p_o be the matrix containing only the stable transfer functions of p_a. Then

$$\Delta p = p_a - p_o \tag{6}$$

contains only the unstable transfer functions of p_a, and therefore in general it is sparse.

Theorem 1. Let Δp be partitioned in the form

$$\Delta p = \begin{bmatrix} \Delta p_1 & \Delta p_2 \\ 0 & \Delta p_4 \end{bmatrix} \tag{7}$$

and we choose $Q \in H$ to have the form

$$Q = \begin{bmatrix} Q_1 & 0 \\ Q_3 & Q_4 \end{bmatrix} \tag{8}$$

Then, $S(p_a,C)$ is stable if $S(\Delta p_1,Q_1)$, $S(\Delta p_4,Q_4)$ and $S(\hat{\Delta p_2},Q_3)$ are stable, where

$$\hat{\Delta p_2} = (1 + \Delta p_1 Q_1)^{-1} \Delta p_2 (1 + Q_4 \Delta p_4)^{-1} \tag{9}$$

and C is given by (5).

Note that the proposed design method represents a big reduction of complexity and this is possible because Δp is sparse and the design is on Q, not on C.

REFERENCES

[1] Q. Huang and R. Liu, "Necessary and Sufficient Condition for Stability of Perturbed Systems," This Journal.

[2] Desoer, C. A. and Liu, R. W., "Global Parameterization of Feedback System with Nonlinear Plants" System and Control Letters, Vol. 1 No. 4, pp. 249-251, Jan. 1982.

[3] A. Bhaya and Desoer, C. A., "Robust Stability Under Additive Perturbations" Memorandum No. UCB/ERL M84/110 7 December 1984.

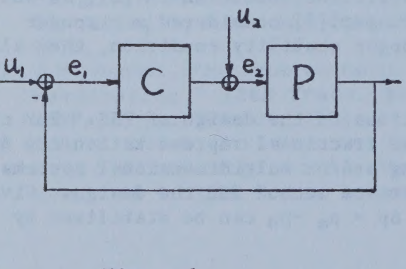


Figure 1

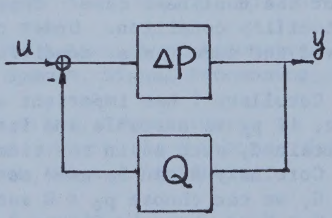


Figure 2

REMARKS ON WEIGHTED SENSITIVITY MINIMIZATION

Antonio M. Pascoal and Pramod P. Khargonekar
Department of Electrical Engineering
University of Minnesota, Minneapolis, MN 55455

ABSTRACT

In this paper we study some inherent limitations existent in the design of SISO feedback control systems to achieve disturbance rejection, by considering an equivalent interpolation problem involving a certain weighted sensitivity function [10].

Explicit formulae are derived to compute the complex weighting function W at each point of its domain given a performance specification function ps, following the technique presented by Young and Francis [9].

1. INTRODUCTION

Consider the linear, time-invariant, single-input single-output (SISO) feedback system of figure 1, with the plant P and controller C represented respectively by the transfer functions P(s) and C(s). The system sensitivity function $S(s) = [1+P(s)C(s)]^{-1}$ plays a key role in the design of feedback control systems (Horowitz [3]), the behavior of its magnitude along the $j\omega$-axis determining the effectiveness of the control system in coping with disturbances d and plant uncertainty, and the ability of the output y to follow closely an input signal u.

With respect to figure 1, we formulate the following problem: given the plant P, derive conditions under which it is possible to find a controller C that stabilizes (internally) the plant and keeps the magnitude of the sensitivity function along the $j\omega$-axis bounded above by a performance specification function $ps(\omega)$, that is $|S(j\omega)| < ps(\omega)$ for all frequencies ω. In the case of the design for disturbance rejection, a possible choice for ps is shown in figure 4 (Young and Francis [9]), and reflects the objective of attenuating by at least ε the low frequency disturbances in the range $[0,\omega_c]$ while keeping an upper bound of δ on the magnitude of the sensitivity function over the complementary range.

In some situations, the design parameters ε, δ and ω_c cannot be specified independently. This stems from the fact that C stabilizes internally the plant P if and only if the sensitivity function S is analytic and bounded in the right half complex plane (an element of the Banach space H_∞) and satisfies certain interpolation constraints that set a lower bound on the value of its H_∞-norm (Zames and Francis [11]). Freundenberg and Looze [2] have shown that the combined right half plane zeros and poles of a scalar plant together with a stabilizing controller impose a design tradeoff, where a decrease in ε can only be achieved by increasing the value of δ. Their results are based on the modification of a Poisson integral-formula (Levinson [5]) to obtain integral relationships that the sensitivity function S must satisfy in order that the feedback system be stable.

In this paper, we study some inherent limitations in the design for disturbance rejection by considering an equivalent interpolation problem involving a certain weigthed sensitivity function (Zames [10]). The

---

This research was supported in part by the NSF under Grant No. ECS-8451519.

complex weighting function W is obtained from the performance specification function $ps(\omega)$ using techniques from complex analysis, as proposed by Young and Francis [9] in their study of sensitivity tradeoffs for MIMO systems. For a certain class of functions $ps(\omega)$, we derive explicit formulae to compute the weighting function at each point of its domain (section 2). In the case of scalar plants, and for ps as depicted in figure 4, these formulae are used to obtain a lower bound for δ given ε, ω_c and the pattern of right half plane zeros and poles of the plant P only. In particular, the estimates of Freudenberg and Looze [2] are retrieved, but we show that in some cases it is possible to obtain better estimates that reflect the influence of the relative position of the right half plane zeros of the plant P (section 3).

Section 4 is a brief comment on the use of weighting functions W that are rational, and how this imposes even stronger limitations on the design for disturbance rejection.

2. WEIGHTED SENSITIVITY MINIMIZATION AND FORMULAE FOR OUTER FUNCTIONS

Let P be a fixed SISO strictly proper plant with closed right half plane zeros $Z_1, Z_2, \ldots, Z_m$ and closed right half plane poles $P_1, P_2, \ldots, P_n$. Assume, for simplicity, that every zero and pole occurs with multiplicity one except possibly the zero located at ∞ and that the plant does not have zeros or poles on the $j\omega$-axis. Suppose that the plant P is inserted in the feedback scheme of figure 1. As is well known (Youla, Bongiorno and Lu [8]), in order for the closed loop system to be internally asymptotically stable it is necessary and sufficient that the sensitivity function $S(s)$ have the following properties:

i) $S(s)$ is analytic in the extended close right half plane $\tilde{H} = \overline{H} \cup \{\infty\}$, where $\overline{H} = \{s \varepsilon \mathbb{C}: \operatorname{Re}(s) \geq 0\}$

ii) the zeros of $S(s)$ contain $\{P_1, P_2, \ldots, P_n\}$, multiplicities included

iii) the zeros of $S(s)-1$ contain $\{Z_1, Z_2, \ldots, Z_m\}$, multiplicities included.

In the design for disturbance rejection, the sensitivity function must satisfy the additional constraint $|S(j\omega)| \leq ps(\omega)$, ω in $(-\infty, +\infty)$, where ps is a performance specification function. Denoting $1/ps(\omega)$ by $V_b(\omega)$, this constraint becomes $\sup|S(j\omega)V_b(\omega)| \leq 1$, $\omega \varepsilon R$. Suppose now that it is possible to find a function $W(s)$, analytic, bounded and devoid of zeros in the open right half plane, and such that $\lim|W(s)|$ when s approaches the $j\omega$-axis non-tangentially is $V_b(\omega)$ (almost everywhere). The additional constraint can then be rewritten as

$$\sup_{\operatorname{Re}(s)>0} |S(s)W(s)| = \|SW\|_\infty = \|S_W\|_\infty \leq 1 \tag{2.1}$$

where $\|\cdot\|_\infty$ denotes the norm in the Banach Space H_∞ of complex valued functions of s which are analytic and bounded in the right half complex plane. In conclusion, the existence of an analytic function $S_W: H \cup \{\infty\} \rightarrow \mathbb{C}$ (defined by $S_W(s) = S(s)W(s)$) satisfying: i) $S_W(P_i) = 0$, $i = 1, 2, \ldots, n$; ii) $S_W(Z_j) = W(Z_j)$, $j = 1, 2, \ldots, m$; iii) $\|S_W\|_\infty \leq 1$ is a necessary condition for the existence of a controller C that internally stabilizes P and enables the overall control system to meet the performance specifications.

For mathematical convenience, this interpolation problem will be formulated on the unit disc $\overline{D} = \{z \varepsilon \mathbb{C}: |z| \leq 1\}$. Let $\phi: \tilde{H} \rightarrow \overline{D}$ be a fixed conformal equivalence. Let $\psi_i = \phi(P_i)$, $i = 1, 2, \ldots, n$ and $\xi_j = \phi(Z_j)$, $j = 1, 2, \ldots, m$. In particular, the points at ∞ will be mapped on the boundary T of the unit disc D. Hence all functions of s are transformed into functions of z via $s = \phi^{-1}(z)$, and the possibility of finding a suitable controller C implies the existence of an analytic function $\tilde{S}: D \to \mathbb{C}$, bounded in D and satisfying

$$\begin{aligned} &\text{i)} \quad \tilde{S}(\psi_i) = 0,\ i = 1, 2, \ldots, n \\ &\text{ii)} \quad \tilde{S}(\xi_j) = W(\phi^{-1}(\xi_j)) = \tilde{W}(\xi_j),\ j = 1, 2, \ldots, q \qquad (2.2) \\ &\text{iii)} \quad \sup_{|z|<1} |\tilde{S}(z)| = \|S\|_\infty \leq 1 \end{aligned}$$

where $q<m$ is the number of finite zeros of P in the open right half plane. We are thus considering only the "interior interpolation data" (Khargonekar and Tannenbaum [4]).

COMPUTATION OF OUTER FUNCTION $\tilde{W}(Z)$

We see that it is important to find $\tilde{W}(z)$ for z in D, given its magnitude along the boundary T of $\overline{D}$. Its computation is based on the following theorem:

THEOREM 1 (Rudin [6]): If $\hat{q}$ is a positive measurable function on T such that $\log q$ is in $L^1(T)$, and if

$$Q(z) = c \exp\left\{ \frac{1}{2\pi} \int_{-\pi}^{+\pi} \frac{e^{j\lambda}+z}{e^{j\lambda}-z} \log\hat{q}(e^{j\lambda})\, d\lambda \right\}, |c| = 1 \qquad (2.3)$$

for z in D, then Q is an OUTER function. Furthermore, i) $\lim_{r\to 1} |Q(re^{j\lambda})| = \hat{q}(e^{j\lambda})$ a.e. on T; ii) Q belongs to H_∞ (Hardy space of complex-valued functions of z which are analytic and bounded on D) if and only if $\hat{q}$ belongs to $L^\infty(T)$. In this case, $\|Q\|_\infty = \|\hat{q}\|_\infty$.

The next theorem gives an explicit formula for $\tilde{W}(z)$ (z in D) when $\hat{q}$ is generated by the performance specification function $ps(\omega)$ represented in figure 2, through the relationship $\hat{q}(e^{j\lambda}) = 1/[ps(\phi^{-1}(e^{j\lambda}))]$, $\lambda \varepsilon [-\pi, +\pi]$.

THEOREM 2: Consider the performance specification function ps defined by

$$ps(\omega) = \begin{cases} \gamma, & \text{for } \omega \text{ in } [-\omega_2, -\omega_1] \cup [\omega_1, \omega_2] \\ 1, & \text{otherwise} \end{cases} \qquad (2.4)$$

Given a fixed conformal mapping $\phi: \tilde{H} \to \overline{D}$, the corresponding weighting function $\tilde{W}(z)$, for z in D equals $|\tilde{W}(z)| \exp[j\Phi(z)]$, where

$$|\tilde{W}(z)| = (1/\gamma)^{\frac{\theta(z)}{\pi}}$$

$$\Phi(z) = (1/2\pi)\log(1/\gamma)\log[\Phi^*(\omega_1,z)/\Phi^*(\omega_2,z)]$$

$$\text{with } \theta(z) = \theta^*[\omega_2,z]-\theta^*[-\omega_2,z]-\theta^*[\omega_1,z]+\theta^*[-\omega_1,z] \qquad (2.5)$$

$$z = \phi(x+jy);\ \theta^*[\omega_i,z] = \tan^{-1}[(\omega_i-y)/x],\ i = 1, 2;\ x+jy \varepsilon H$$

$$\Phi^*(\omega_i,z) = [x^2+(\omega_i+y)^2]/[x^2+(\omega_i-y)^2]$$

Proof: $\tilde{W}(z) = c \exp[a(z)+jb(z)]$ where, from (2.3),

$$a(z) = \frac{1}{2\pi} \int_{-\pi}^{+\pi} \mathrm{Re}\left\{ \frac{e^{j\lambda}+z}{e^{j\lambda}-z} \right\} \log\hat{q}(e^{j\lambda})d\lambda \tag{2.6}$$

and $\log\hat{q}(e^{j\lambda}) = \log V_b[\phi^{-1}(e^{j\lambda})]$ (see fig. 3).

To compute $a(z)$ at a particular point, say $z=\xi$, consider a bilinear transformation

$$\eta: \overline{D}\to\overline{D},\ \eta(z) = \beta\cdot(z-\xi)/(1-\overline{\xi}z), |\beta| = 1 \tag{2.7}$$

In particular, $\eta(\xi)=0$. The integral in (2.6) can be evaluated as $(1/2\pi)\cdot \int_{-\pi}^{+\pi} \log q^*(e^{i\lambda})d\lambda$, where $q^*(e^{i\lambda})=\hat{q}(\eta^{-1}(e^{i\lambda}))$ (Ahlfors [1]). Due to the simple form of the function $\log \hat{q}(e^{i\lambda})$, this integral is easily computed as $\log(1/\gamma)\cdot T$, where T is the total length of the images of the segments $[-j\omega_2,-j\omega_1]$ and $[j\omega_1,j\omega_2]$ through the mapping $\eta o\phi$. The constant β in (2.7) can always be chosen so that $\eta o\phi(s)=-(\overline{s}_0/s_0)\cdot(s-s_0)/(s+\overline{s}_0)$ $(s_0=\phi^{-1}(\xi))$, a bilinear transformation mapping the right half complex plane into the unit disc, carrying $\phi^{-1}(\xi)$ to the origin and the point 0 to 1. Denoting $\phi^{-1}(\xi)$ by $x+jy$, and noticing that $\eta o\phi(j\omega)=\exp\{-2\tan^{-1}(y/x)-2\tan^{-1}[(\omega-y)/x]\}$, $\omega\varepsilon R$, the expression for $|\tilde{W}(z)|$ follows immediately; the phase of $\tilde{W}(z)$ is computed by noticing that $a(z)$ and $b(z)$ are harmonic conjugate functions on the unit disc D, related through the Cauchy-Riemann equations (Levinson [5]). Integrating these equations and requiring the phase b to be zero for $z=\phi(0)$, the expression for $\Phi(z)$ is obtained.□

Suppose that the performance specification function ps can be expressed as the pointwise product of k functions ps_i of the type depicted in fig. 2, i.e., $ps(\omega) = \prod_{i=1}^{k} ps_i(\omega)$. In this case, the corresponding weighting function $\tilde{W}$ can be compute as $\tilde{W}(z) = \prod_{i=1}^{k} \tilde{W}_i(z)$, where each $\tilde{W}_i(Z)$ is calculated using the formulae given in (2.5). In particular, and for ps as in figure 4, given a fixed conformal mapping ϕ: $\overline{H}\to\overline{D}$ we obtain

$$|\tilde{W}(z)| = (1/\varepsilon)^{\frac{\theta(z)}{\pi}} (1/\delta)^{\frac{\pi-\theta(z)}{\pi}} \tag{2.8}$$

with

$$z = \phi(x+jy);\ \theta(z) = \tan^{-1}[(\omega_c+y)/x]+\tan^{-1}[(\omega_c-y)/x]$$

3. DESIGN TRADEOFFS (DISTURBANCE REJECTION)

The existence of an analytic function $\tilde{S}$: $D\to\mathbb{C}$ satisfying conditions (2.2) is now discussed. Define the following interpolation data:

$$\begin{aligned} &a_i = \xi_i \text{ in } D;\ b_i = \tilde{W}(\xi_i),\ i = 1, 2, \ldots, q \\ &a_{j+q} = \psi_j \text{ in } D;\ b_{j+q} = 0,\ j = 1, 2, \ldots, n. \end{aligned} \tag{3.1}$$

where it is assumed that the set of points a_i, i = 1, 2, ..., q+n does not contain repeated values. The function $\tilde{S}$, analytic on D and with norm $\|S\|_\infty<1$ satisfies the interpolation constraints if and only if it can be written as

$$\tilde{S}(z) = \prod_{j=1}^{n} [(z-a_{j+q})/(1-\bar{a}_{j+q}z)]\cdot\tilde{S}_1(z) = B_P(z)\cdot\tilde{S}_1(z) \tag{3.2}$$

where $B_P(z)$ denotes a Blaschke product and $\tilde{S}_1$: D→D is analytic, with norm $\|\tilde{S}_1\|_\infty<1$ and satisfies the interpolation constraints (Walsh [7])

$$\tilde{S}_1(a_i) = B_P^{-1}(a_i)\cdot\tilde{S}(a_i) = B_P^{-1}(a_i)\cdot b_i,\ i = 1, 2, \ldots, q \tag{3.3}$$

This is possible only if $|\tilde{S}_1(a_i)|<1$, i = 1, 2, ..., q or, using (3.3)

$$|b_i| < |B_P(a_i)|,\ i = 1, 2, \ldots, q \tag{3.4}$$

$|B_P(a_i)|$ is the product of n terms of the form $\mu(a_i,a_{j+q}) = |a_i-a_{j+q}|/|1-\bar{a}_{j+q}a_i|$, j = 1, 2, ..., n where $\mu(a_i,a_{j+q})$ is a conformal invariant in the following sense (Khargonekar and Tannenbaum [4]): given β: D→D a conformal equivalence, $\mu(a_i,a_{j+q}) = \mu(\beta(a_i),\beta(a_{j+q}))$. In particular, selecting β so that one of the points (a_i or a_{j+q}) is mapped through β into the origin, it follows easily that $\mu(a_i,a_{j+q}) = |Z_i-P_j|/|Z_i+\bar{P}_j|$ with $a_i = \xi_i = \phi(Z_i)$, i = 1, 2, ..., q; $a_{j+q} = \psi_j = \phi(P_j)$, j = 1, 2, ..., n and ϕ: $\tilde{H}\to\tilde{D}$ the fixed conformal mapping used in (2.2). $|B_p(a_i)|$, i = 1, 2, ..., q is thus expressed directly in terms of the zero Z_i and poles P_1, P_2, ..., P_n of the plant P in the right half complex plane. Suppose that the performance specification function ps is the one depicted in figure 4. A set of necessary conditions for the existence of a controller C stabilizing the plant P and meeting the performance specifications is obtained from (3.4) by substituting the values for $|B_p(a_i)|$ and $|b_i|=|\tilde{W}(\xi_i)|$, i = 1, 2, ..., q as given in (2.8):

$$(1/\varepsilon)^{\frac{\theta_i}{\pi}}\,(1/\delta)^{\frac{\pi-\theta_i}{\pi}} < \prod_{j=1}^{n} |Z_i-P_j|/|Z_i+\bar{P}_j| = |B_p(Z_i)| = |B_p(a_i)| \tag{3.5}$$

where

$$Z_i = x_i+jy_i;\ i = 1, 2, \ldots, q$$

$$\theta_i = \tan^{-1}[(\omega_c+y_i)/x_i]+\tan^{-1}[(\omega_c-y_i)/x_i]$$

and $\{Z_i\}$ and $\{P_j\}$ represent respectively the set of finite zeros and poles of P on the right half complex plane H. Solving explicitly for δ,

$$\delta > \delta_F = \max_i \delta_i,\ \delta_i = (1/\varepsilon)^{\frac{\alpha_i}{1-\alpha_i}}\,|B_p^{-1}(Z_i)|^{\frac{1}{1-\alpha_i}} \tag{3.6}$$

$$i = 1, 2, \ldots, q;\ \alpha_i = \theta_i/\pi$$

where θ_i and $B_p(Z_i)$ are computed as in (3.5).

For fixed values of ε and ω_c, and for a given configuration of right half plane poles and zeros of the plant P, this expression gives a lower bound on δ, the norm of the sensitivity function S. This is exactly the estimate derived by Freudenberg and Looze [2] taking in consideration the zeros and poles of the plant P only.

Using the theory of interpolation, it is possible to obtain a lower bound for δ that reflects the relative position of the right half plane zeros of P, as we illustrate with the next theorem.

THEOREM 3: Consider a plant P with two distinct real, finite right half plane zeros Z_1 and Z_2 $(Z_2 > Z_1)$ and any finite number of distinct right half plane poles $P_1, P_2, \ldots, P_n$ (possibly complex). Suppose that the number of real poles located between Z_1 and Z_2 is odd. Then, it is a necessary condition for the existence of a controller C that stabilizes the plant P while meeting the performance specification represented in fig. 4 that

$$\delta \geqslant \delta_Z = \max_i \, (1/\varepsilon)^{\frac{\alpha_i}{1-\alpha_i}} \, |B_p^{-1}(Z_i)|^{\frac{1}{1-\alpha_i}} \, |B_Z^{-1}|^{\frac{1}{1-\alpha_i}} \; ; \; i = 1, 2 \tag{3.7}$$

where α_i, $B_p(Z_i)$ are as in (3.5) and (3.6), and $|B_Z| = |Z_1 - Z_2|/|Z_1 + Z_2|$ shows the influence of the relative location of the zeros Z_1 and Z_2.

Proof: Following the procedure used at the beginning of this section, and using the notation introduced in (3.1), it is required to find $\tilde{S}_1 : D \to D$ analytic, with norm $\|\tilde{S}_1\|_\infty \leqslant 1$, satisfying the interpolation constraints $\tilde{S}_1(a_i) = b_i \, B_p^{-1}(a_i) = b_i^*$; $i = 1, 2$ with $|b_i^*| \leqslant 1$ (see 3.4). $B_p(a_i)$, defined in (3.2), is easily computed by selecting the conformal mapping $\phi : \tilde{H} \to \overline{D}$ as $\phi(s) = (1-s)/(1+s)$, to give $B_p(a_i) = \prod_{j=1}^{n} [(Z_i - P_j)/(Z_i + \overline{P}_j)][-(1+\overline{P}_j)/(1+P_j)]$. The existence of $\tilde{S}_1$ is equivalent to the existence of a function $\tilde{S}_2 : D \to D$, analytic with norm $\|\tilde{S}_2\|_\infty \leqslant 1$, defined by (Walsh [7])

$$\tilde{S}_2(z) = \frac{\tilde{S}_1(z) - b_1^*}{1 - \overline{b}_1^* \tilde{S}_1(z)} \cdot \frac{1 - \overline{a}_1 z}{z - a_1}$$

where $\tilde{S}_2$ satisfies the interpolation constraint $\tilde{S}_2(a_2) = (b_2^* - b_1^*)(1 - \overline{a}_1 a_2)/(1 - \overline{b}_1^* b_2^*)(a_2 - a_1) = b_2^{**}$. A necessary condition for the existence of such a function is that $|b_2^{**}| \leqslant 1$, or,

$$|(b_2^* - b_1^*)/(1 - \overline{b}_1^* b_2^*)| \leqslant |(a_2 - a_1)/(1 - \overline{a}_1 a_2)| = |B_Z| = |(Z_2 - Z_1)/(Z_2 + Z_1)| \tag{3.8}$$

where $|B_Z|$ is a conformal invariant; b_1 and b_2 defined in (3.1) are both real, as well as $B_p(a_i)$, $i = 1, 2$ since complex poles occur in conjugate pairs. If the number of real poles located between Z_1 and Z_2 is odd, b_1^* and b_2^* will have different signs, and (3.8) can be rewritten as $|b_2^*|(|b_1^*| - 1/|B_Z|) + (1 - |b_1^*|/|B_Z| \geqslant 0$ after factoring out $|b_2^*|$. Since $(|b_1^*| - 1/|B_Z|) \leqslant 0$, we obtain that $|b_1^*| \leqslant |B_Z|$ is a necessary condition for the interpolation to be possible. Factoring out $|b_1^*|$, the other necessary condition is $|b_2^*| \leqslant |B_Z|$. The result follows by substituting in these two inequalities the expressions for b_i^* and $|B_Z|$ and solving explicitly for δ. □

The theorem can be extended to the case where there are more than two real right half plane zeros using the same type of technique.

4. RATIONAL WEIGTHING FUNCTIONS

In the design for disturbance rejection, the controller transfer function C(s) can be recovered from the weigthed sensitivity function $S(s)W(s) = [1 + P(s)C(s)]^{-1} W(s)$ satisfying the required interpolation constraints. To simplify the structure of the controller, it is convenient to modify the performance specification function in such a way as to generate an outer function W that is rational. This procedure will introduce more limitations on the performance achievable, specially when W has a low order McMillan degree. As an example, consider the case of a plant P with a single real, finite right half plane zero Z_1 and the performance specification ps_r depicted in fig. 5 (generating the weighting function $W(s) = [\delta\cdot(s+\omega_c)/(s+\omega_c\delta/\varepsilon)]^{-1}$).

A necessary condition for the existence of a controller C that stabilizes P and meets the performance specifications is $|W(Z_1)|<1$, or using the expression for W(s), $\delta[Z_1-\omega_c(1-\varepsilon)/\varepsilon]>Z_1$, where $Z_1>0$. This equation is satisfied only if $\omega_c < \omega_c^* = Z_1\cdot\varepsilon/(1-\varepsilon)$, which imposes an upper bound on the length of the interval $[0,\omega_c]$ over which disturbance attenuation (by the factor of ε) can be achieved. For $\omega_c<\omega_c^*$, δ must satisfy $\delta > Z_1/[Z_1-\omega_c(1-\varepsilon)/\varepsilon]$, and as ω_c approaches ω_c^*, δ becomes arbitarily large.

REFERENCES

[1] Lars V. Ahlfors, Complex Analysis, McGraw-Hill, 1953.

[2] J. S. Freudenberg and D. P. Douglas, Right Half Plane Poles and Zeros and Design Tradeoffs in Feedback Systems, IEEE Trans. on Auto. Control, Vol. AC-30, No. 6, June 1985, pp. 555-565.

[3] I. M. Horowitz, Synthesis of Feedback Systems, Academic Press, New York, NY, 1963.

[4] P. P. Khargonekar and A. Tannenbaum, Noneuclidean Metrics and Robust Stabilization of Systems with Parametric Uncertainty, to appear in the IEEE Trans. on Auto. Control, October 1985.

[5] N. Levinson and R. M. Redheffer, Complex Variables, San Francisco, CA: Holden-Day, 1970.

[6] W. Rudin, Real and Complex Analysis, McGraw-Hill, New York, 1966.

[7] J. Walsh, Interpolation and Approximation by Rational Functions in the Complex Domain, AMS Colloquium Publications, 20, Fourth Edition, 1965.

[8] D. C. Youla, J. Bongiorno and Y. Lu, Single Loop Feedback Stabilization of Linear Multivariable Dynamic Plants, Automatica, 10: 159-173, 1974.

[9] S. D. Young and B. A. Francis, Sensitivity Tradeoffs for Multivariable Plants, IEEE Trans. on Auto. Control, Vol. AC-30, No. 7, July 1985, pp. 625-632.

[10] G. Zames, Feedback and Optimal Sensitivity: Model Reference Transformations, Multiplicative Seminorms, and Approximate Inverses, IEEE Trans. on Auto. Control, Vol. AC-26, No. 2, April 1981, pp. 301-320.

[11] G. Zames and B. A. Francis, Feedback, Minimax Sensitivity and Optimal Robustness, IEEE Trans. on Auto. Control, Vol. AC-28, No. 5, May 1983, pp. 585-600.

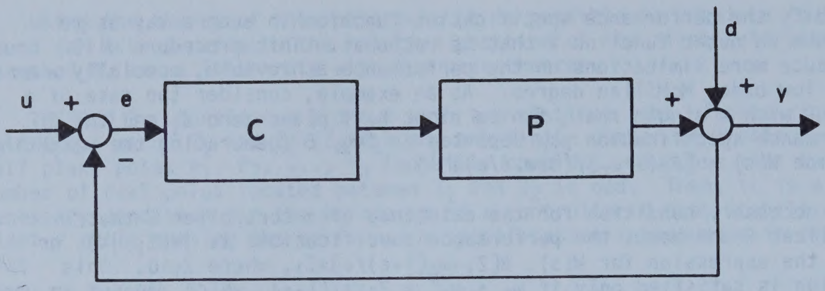


figure 1

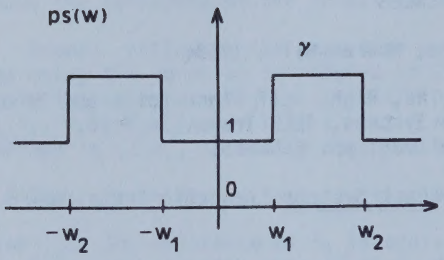


figure 2

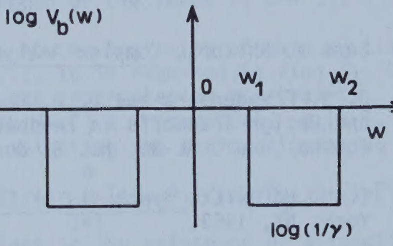


figure 3

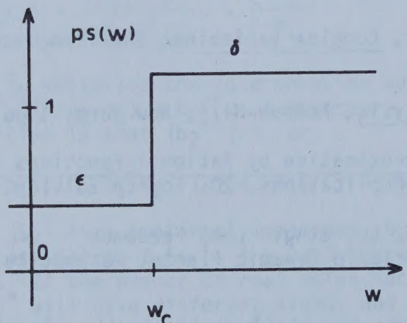


figure 4

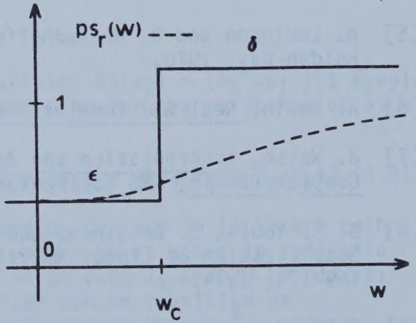


figure 5

On the problem of simultaneously stabilizing a one parameter family of single input single output linear dynamical systems.

BIJOY K. GHOSH
Department of Systems Science and Mathematics
Washington University
Saint Louis, MO 63130

Abstract: In this paper we investigate the problem of simultaneously stabilizing a linear one parameter family of plants with single input and single output by a non-switching dynamic compensator.

Introduction: Let H be the ring of proper rational functions with real coefficients with poles in the open left half of the complex plane. Let $x_1(s)/y_1(s)$, $x_2(s)/y_2(s)$; $x_1, x_2, y_1, y_2 \in H$ be a pair of distinct single input single output plants of degree n. For ease of exposition and notational simplicity we assume that the zeros of $x_1 y_2 - x_2 y_1$ in C^+ (the closed right half of the complex plane) are simple and are in the interior of C^+. Let us denote these zeros by $a_1, \ldots, a_t$. Let us now define b_i as follows:

$$b_i = \begin{array}{ll} x_2(a_i)/x_1(a_i) & \text{if the multiplicity of } a_i \text{ as a common zero of } x_1(s) \text{ and } x_2(s) \text{ is less than or equal to the multiplicity of } a_i \text{ as a common zero of } y_1(s) \text{ and } y_2(s). \\ y_2(a_i)/y_1(a_i) & \text{otherwise.} \end{array}$$

for $i=1, \ldots, t$. Let us also define $s_i = (a_i - 1)/(a_i + 1)$ and $z_i = (\sqrt{b_i} - 1)/(\sqrt{b_i} + 1)$ where the branch cut for the square root is taken to be the non-positive real axis. Furthermore let k be the largest real number such that

$$[\,1 - k^2 z_i \, z_j\,] / [\,1 - s_i \, s_j\,]_{i,j=1}^{t}$$

is non-negative definite.

Now consider a linear one parameter family of plants described as follows.

$$F = \left\{ \begin{array}{l} g_\$ (s) : g_\$ = [\$ x_1 + (1-\$) x_2]/[\$ y_1 + (1-\$) y_2], \\ \$ \in [0,1], \ \deg g_\$ = n \text{ for all } \$. \end{array} \right\}$$

Then we have the following main result.

Main Theorem: There exists a non-switching proper compensator which simultaneously stabilizes each and every plant in F iff k is strictly greater than 1.

Proof of the main theorem: Using [2, corollary 6.2] and defining the appropriate conformal transformations, it follows that the family F is simultaneously stabilizable by a proper compensator iff there exists a rational function f(s) with real co-efficients analytic on $|s| < 1$ and continuous on $|s| \leq 1$ such that $f(s_i) = z_i$, $i=1, \ldots, t$ and the image of the closed unit disc under f does not intersect the unit circle. If $k > 1$ it follows from the standard Nevanlinna-Pick interpolation theory (see

[1, proposition 2.1]in particular) that there exists f(s) satisfying the above interpolation constraints and such that $\sup[\,|f(s)| : |s| \leqslant 1] \leqslant 1/k$. Conversely if k=1, for every f(s) satisfying the above interpolation constraint, the image of the closed unit disc intersects the unit circle. The details are omitted. (Q.E.D)

Remark 1: The proof of the Main Theorem uses the general idea of conformal equivalence as described by Khargonekar and Tannenbaum [4] and the interpolation condition of [2, theorem 6.1].

Remark 2: As noted in [2], the gain margin problem considered in [4] follows as a corollary of the main theorem.

Remark 3: We find it surprising to note that the Nevanlinna-Pick interpolation theory completely settles the simultaneous stabilization problem of a one parameter linear family of plants described by F.

Remark 4: Generalizing the main theorm to a linear 2-parameter family of plants would be a non-trivial proposition. In fact it is known [2, theorem 6.1] (also note in [4] the comment on simultaneous pole zero variation) that for a 2-parameter family of plants one obtains a more difficult interpolation problem defined in [3] as the transcendental problem.

References:

[1] B.K. Ghosh,"Simultaneous stabilization, sensitivity minimization, tracking and disturbance rejection by interpolation methods," 22nd Annual Allerton Conference on Communication, Control and Computing, 1984, pp. 625-634.

[2] B.K. Ghosh,"Simultaneous partial pole placement — A new approach to multimode system design," To appear in the IEEE Trans. Aut. Contr.

[3] B.K. Ghosh,"Transcendental and interpolation methods in simultaneous stabilization and simultaneous partial pole placement problems," To appear in SIAM J. on Control and Optimization.

[4] P.P. Khargonekar and A. Tannenbaum,"Non-Euclidean metrics and the robust stabilization of systems with parameter uncertainty," IEEE Trans. Aut. Contr., AC-30, 10, Oct 1985, pp. 1005-1013.

INNER-OUTER FACTORIZATIONS ARISING IN H^∞ OPTIMIZATION PROBLEMS

B-C Chang
Department of Electrical and Computer Engineering and Technology
Bradley University
Peoria, Illinois 61625

ABSTRACT

The inner-outer factorization algorithm proposed by Chu and Doyle is numerically reliable and requires less computation than any other existing method. Unfortunately, this algorithm cannot be applied directly to most of the realistic problems because the restrictive assumptions on which the algorithm is developed are not usually satisfied. In this paper, an extraction technique is used to decompose a given rational matrix into two parts, one is outer and the other satisfies the above assumptions such that the Chu and Doyle's algorithm can be used.

I. INTRODUCTION

Inner-outer factorizations are essential in obtaining solutions to H^∞ optimization problems [1,2,3,4]. In this paper, the inner-outer factorizations arising in H^∞ optimization problems will be considered.

The H^∞ optimization problem is:

Find a $Z(s) \in R(s)^{m\times r}$ which is analytic in the right half plane such that $\|\Phi\|_\infty$ is minimized where $\Phi(s) = H(s) + S(s)Z(s)T(s)$ with $H(s) \in (RH^\infty)^{p\times q}$, $S(s) \in (RH^\infty)^{p\times m}$, and $T(s) \in (RH^\infty)^{r\times q}$ and $S(s)$ and $T(s)$ having no zeros on the $j\omega$-axis. (1-1)

Here $R(s)^{p\times m}$ ($R[s]^{p\times m}$) denotes the set of pxm rational matrices (polynomial matrices) with real coefficients and $(RH^\infty)^{p\times m}$ is a subset of $R(s)^{p\times m}$ with entries in H^∞ which is the space of all functions that are analytic and bounded in the right half plane. $\|\Phi\|_\infty$ is defined by

$$\| \Phi \|_\infty := \sup_\omega \bar{\sigma}\, \Phi\, [(j\omega)] \tag{1-2}$$

and $\bar{\sigma}\, [\Phi(j\omega)]$ is the largest singular value of $\Phi(j\omega)$.

The first step in solving the above problem is to find the inner-outer factorizations for S(s) and T(s). According to the dimensions of S(s) and T(s) we have the following cases to consider: (a) $p \le m$, $r \ge q$; (b) $p > m$, $r < q$; (c) $p \le m$, $r < q$; and (d) $p > m$, $r \ge q$.

In case (a), S(s) and T(s) are assumed to be of row rank and of column rank respectively and therefore they can be decomposed as

$$S(s) = S_i(s)S_o(s), \quad T(s) = T_o(s)T_i(s) \tag{1-3}$$

where $S_i(s) \in (RH^\infty)^{p\times p}$ is inner, $T_i(s) \in (RH^\infty)^{q\times q}$ is co-inner, $S_o(s) \in (RH^\infty)^{p\times m}$ has its right inverse analytic in the right half plane, and $T_o(s) \in (RH^\infty)^{r\times q}$ has its left inverse analytic in the right half plane. $S_o(s)$ and $T_o(s)$ are called outer matrices. A rational matrix $U(s) \in (RH^\infty)^{p\times m}$ is inner if $U^*(j\omega)U(j\omega) = I$ for all ω and co-inner if $U(j\omega)U^*(j\omega) = I$ for all ω. A^* is the conjugate transpose of the constant matrix A. We will also use A^* as a short-hand notation for $A^T(-s)$ when A(s) is a rational matrix. For notational simplicity, we will omit the indeterminate s when there is no confusion.

Define $\hat{Z} = S_oZT_o$ and $\hat{\Phi} = S_i^*\Phi T_i^*$. Then we have

$$\hat{\Phi} = S_i^* H T_i^* + \hat{Z} \tag{1-4}$$

Note that $\|\hat{\Phi}\|_\infty = \|\Phi\|_\infty$ and therefore case (a) of problem (1-1) is equivalent to the following:

This research was supported by National Science Foundation under Grant No. ECS-8404143.

Find a $\hat{Z}(s) \in (RH^\infty)^{p \times q}$ such that $||\hat{\Phi}||_\infty$ is minimized where $\hat{\Phi}(s)$ is given by (1-4). (1-5)

Problem (1-5) is referred as the standard H^∞ optimization problem. The standard H^∞ optimization problem was solved by Zames, Francis, and Helton [2], Chang and Pearson [3], Safonov and Verma [5] and Glover [6].

In case (b), S and T are thin and fat respectively, i.e., $p > m$ and $r < q$ and they can be decomposed as

$$S(s) = S_i(s)S_o(s), \quad T(s) = T_o(s)T_i(s) \tag{1-6}$$

where $S_i \in (RH^\infty)^{p \times m}$ is inner, $T_i \in (RH^\infty)^{r \times q}$ is co-inner, $S_o \in (RH^\infty)^{m \times m}$ and $T_o \in (RH^\infty)^{r \times r}$ are outer matrices. Furthermore, we can find a $S_\perp(s) \in (RH^\infty)^{p \times (p-m)}$ such that $[\ S_i \ \ S_\perp\ ]$ is square and inner. Similarly, $T_\perp(s) \in (RH^\infty)^{(q-r) \times q}$ can be found such that $[T_i^T \ T_\perp^T]^T$ is square and co-inner. Now, (1-1) can be rewritten as

$$\Phi = H + [S_i \quad S_\perp] \begin{bmatrix} \hat{Z} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} T_i \\ T_\perp \end{bmatrix} \tag{1-7}$$

here $\hat{Z} = S_o Z T_o$. Define $\hat{\Phi} = [\ S_i \ \ S_\perp]^* \ \Phi\ [\ T_i^* \ \ T_\perp^*]$, then we have

$$\hat{\Phi} = \begin{bmatrix} S_i^* H T_i^* + \hat{Z} & S_i^* H T_\perp^* \\ S_\perp^* H T_i^* & S_\perp^* H T_\perp^* \end{bmatrix} := \begin{bmatrix} H_{11} + \hat{Z} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \tag{1-8}$$

Note that $||\hat{\Phi}||_\infty = ||\Phi||_\infty$ and therefore case (b) of problem (1-1) is equivalent to the following:

Find a $\hat{Z}(s) \in (RH^\infty)^{m \times r}$ such that $||\hat{\Phi}||_\infty$ is minimized where $\hat{\Phi}$ is given by (1-8). (1-9)

Problem (1-9) is referred to as the general H^∞ optimization problem. An iterative type algorithm was proposed by Doyle [7] for solving the general H^∞ optimization problem. This algorithm was simplified by Chang and Pearson [8] and Chu and Doyle [9].

Cases (c) and (d) of problem (1-1) can be treated in a similar way. They can be transformed into a problem like problem (1-9) and the corresponding $\hat{\Phi}(s)$ is a special case of (1-8) with either $[H_{21} \ \ H_{22}] = 0$ or $[\ H_{12}^T \ \ H_{22}^T\ ] = 0$.

It is easy to see the inner-outer factorization of T(s) is just the dual of that of S(s). Therefore, in this paper only the inner-outer factorizations of S(s) will be considered. Recall that $S(s) \in (RH^\infty)^{p \times m}$. There are two cases of inner-outer factorizations of S(s) arising in H^∞ optimization problems. The first case is:

Given $S(s) \in (RH^\infty)^{p \times m}$ which has no zeros on jω-axis and $p \leq m$, find a square inner matrix $S_i(s) \in (RH^\infty)^{p \times p}$ and an outer matrix $S_o(s) \in (RH^\infty)^{p \times m}$ such that $S(s) = S_i(s)\ S_o(s)$. (1-10)

On the other hand, S(s) may be of column rank and $p \geq m$. In this case, the problem is:

Given $S(s) \in (RH^\infty)^{p \times m}$ which has no zeros on jω-axis and $p \geq m$, find an inner matrix $S_i(s) \in (RH^\infty)^{p \times m}$, a complementary inner factor $S_\perp(s) \in (RH^\infty)^{p \times (p-m)}$, and an outer matrix $S_o(s) \in (RH^\infty)^{m \times m}$ such that

$$S(s) = [S_i(s) \quad S_\perp(s)] \begin{bmatrix} S_o(s) \\ 0 \end{bmatrix} \tag{1-11}$$

and $[\ S_i(s) \ \ S_\perp(s)\ ]$ is inner.

Problems (1-10) and (1-11) were considered respectively by Chang and Pearson [10] and Ganesh and Pearson [11]. Both of them use the Bauer algorithm to compute the inner factors. Chu and Doyle [12] proposed a new approach to solve a special case of problem (1-11) with the restriction that S(s) is of column rank at $s = \infty$. The restrictions that $p \geq m$ and S(s) is of column rank at $s = \infty$ are not satisfied in most realistic control problems.

The basic idea of our approach for solving problems (1-10) and (1-11) is the introduction of a preliminary factorization. Decompose S(s) into two factors: the right factor is outer and the left factor is thin or square and of column rank at $s = \infty$. Then Chu and Doyle's algorithm can be used for the inner-outer factorization of the left factor.

In section II, we will review YJB controller parametrization and Chu and Doyle's results. In section III, an extraction technique is presented to take care of the cases where $p < m$ and S(s) has zeros at $s = \infty$.

II. BACKGROUND MATERIAL

Consider the system

$$\begin{bmatrix} z(s) \\ y(s) \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \begin{bmatrix} e(s) \\ u(s) \end{bmatrix} \tag{2-1}$$

where $G_{11}(s)$, $G_{12}(s)$, $G_{21}(s)$, $G_{22}(s)$ are proper rational matrices with real coefficients. The vectors y(s), z(s), u(s), and e(s) are measured output, controlled output, control input, and exogenous input respectively. The measured output vector y(s) contains all the variables that can be measured. The entries of z(s) are the variables to be controlled. The exogenous input vector e(s) consists of the reference inputs and the disturbance inputs. The vectors y(s) and z(s) may have some components in common and z(s) may contain the weighted control inputs.

The exogenous input e(s) can be modeled as

$$e(s) = W(s)v(s) \tag{2-2}$$

where v(s) is any vector with its entries in the Hardy space H^2 (i.e., v(s) is any signal vector with finite energy) and W(s) is a proper rational matrix with real coefficients which is chosen to reflect the set of disturbances to be attenuated and/or the reference inputs to be tracked. W(s) is not necessarily stable. If W(s) is unstable, then some regulator design techniques [13,14] should be used to make $\Phi(s)$ stable, where $\Phi(s)$ is defined by $z(s) = \Phi(s)v(s)$. In this paper, we assume W(s) is stable and can be absorbed by $G_{11}(s)$ and $G_{21}(s)$. Then (2-1) and (2-2) can be combined as

$$\begin{bmatrix} z(s) \\ y(s) \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \begin{bmatrix} v(s) \\ u(s) \end{bmatrix} := G(s) \begin{bmatrix} v(s) \\ u(s) \end{bmatrix} \tag{2-3}$$

where $G_{11}(s) \in R(s)^{p\times q}$, $G_{12}(s) \in R(s)^{p\times m}$, $G_{21}(s) \in R(s)^{r\times q}$, and $G_{22}(s) \in R(s)^{r\times m}$.

Our objective is to find a proper controller Q(s) which satisfies

$$u(s) = Q(s)y(s) \tag{2-4}$$

such that the closed-loop system is internally stable and $||\Phi||_\infty$ is minimized or asymptotically minimized (to be explained later) where

$$\Phi(s) = G_{11}(s) + G_{12}(s)\, Q(s)\, [I - G_{22}(s)\, Q(s)\,]^{-1}\, G_{21}(s) \tag{2-5}$$

Let

$$G(s) = \{\, A,\, [\, B_1 \;\; B_2\,],\, \begin{bmatrix} C_1 \\ C_2 \end{bmatrix},\, \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix} \} \tag{2-6}$$

be a minimal realization of G(s). Here G(s) = { A, B, C, D } implies a state-space realization and $G(s) = D+C(sI - A)^{-1}B$. Suppose the realization { A, B_2, C_2, D_{22} } is stabilizable and detectable, then the stabilization problem can be solved by using YJB controller parametrization [15,16,4]. Parameterize the controller Q(s) by

$$\begin{bmatrix} u(s) \\ w(s) \end{bmatrix} = \begin{bmatrix} J_{11}(s) & J_{12}(s) \\ J_{21}(s) & J_{22}(s) \end{bmatrix} \begin{bmatrix} y(s) \\ t(s) \end{bmatrix} := J(s) \begin{bmatrix} y(s) \\ t(s) \end{bmatrix} \qquad (2\text{-}7a)$$

$$t(s) = Z(s)\, w(s) \qquad (2\text{-}7b)$$

where

$$J = \{ A+B_2F+KC_2+KD_{22}F, [\, -K \;\; B_2+KD_{22} \,], \begin{bmatrix} F \\ -(C_2+D_{22}F) \end{bmatrix}, \begin{bmatrix} 0 & I \\ I & -D_{22} \end{bmatrix} \} \qquad (2\text{-}7c)$$

and F and K are chosen such that eigenvalues of $A+B_2F$ and $A+KC_2$ are stable. Then the set of proper stabilizing controllers can be described by [4]

$$\{ Q = J_{11} + J_{12}Z[I-J_{22}Z]^{-1} J_{21} : Z \in RH^{\infty},\ I+D_{22}Z(\infty) \text{ invertible} \} \qquad (2\text{-}7d)$$

By using this parametrization, $\Phi(s)$ can be rewritten as [4]

$$\Phi(s) = H(s) + S(s)Z(s)T(s) \qquad (2\text{-}8a)$$

where

$$H(s) = \{ \begin{bmatrix} A+B_2F & -B_2F \\ 0 & A+KC_2 \end{bmatrix}, \begin{bmatrix} B_1 \\ B_1+KD_{21} \end{bmatrix}, [\, C_1+D_{12}F \;\; -D_{12}F \,], D_{11} \} \qquad (2\text{-}8b)$$

$$S(s) = \{ A+B_2F, B_2, C_1+D_{12}F, D_{12} \} \qquad (2\text{-}8c)$$

$$T(s) = \{ A+KC_2, B_1+KD_{21}, C_2, D_{21} \} \qquad (2\text{-}8d)$$

Now the problem is to find a $Z(s) \in (RH^{\infty})^{m\times r}$ such that $||\Phi||_{\infty}$ is minimized or asymptotically minimized. If S(s) and T(s) have no zeros at $s=\infty$, then there exists a $Z(s) \in (RH^{\infty})^{m\times r}$ such that $||\Phi||_{\infty}$ is minimized. Otherwise, we can only find a $Z(s) \in R(s)^{m\times r}$ which is analytic in the right half plane such that $||\Phi||_{\infty}$ is minimized. In this case, the optimal Z(s) is not proper. This difficulty can be removed by using Zames' identity sequence multiplication technique [1]. That is, define

$$Z_n(s) = Z(s)\, [\, n/(s+n) \,]^k \qquad (2\text{-}9)$$

where k is the smallest integer chosen such that $Z_n(s)$ is proper. Define

$$\Phi_n(s) = H(s) + S(s)Z_n(s)T(s) \qquad (2\text{-}10)$$

Then if $\bar{\sigma}\,[H(\infty)] \leq ||\Phi||_{\infty}$, we have

$$\lim_{n \to \infty} \| \Phi_n \|_{\infty} = \| \Phi \|_{\infty} \qquad (2\text{-}11)$$

In this case, we say that $Z_n(s)$ asymptotically minimizes $||\Phi_n||_{\infty}$.

As we mentioned in section I, the first step to solving problem (1-1) is to find the inner-outer factorizations of S(s) and T(s). In this paper, only the inner-outer factorization of S(s) will be considered. The inner-outer factorization of T(s) is just a dual of that of S(s).

$S(s) \in (RH^{\infty})^{p\times m}$ is given by (2-8c) where

$$\{ A, B_2, C_1, D_{12} \} = G_{12}(s) \qquad (2\text{-}12)$$

and F can be any constant matrix such that $A+B_2F$ is stable. If $p \geq m$ and rank $D_{12} = m$, then the following Chu and Doyle's results can be used to compute the inner factor, the outer factor, and the complementary inner factor for S(s).

Theorem 2-1: [4,12]

Given $\hat{S}(s) = \{ \hat{A}, \hat{B}, \hat{C}, \hat{D} \} \varepsilon (RH^{\infty})^{pxm}$ is minimal. If $p \geq m$ and $\hat{S}^{*}\hat{S} > 0$ on the $j\omega$-axis, including at ∞, then there exists a $\hat{F}$ such that

$$\hat{S}_i(s) = \{ \hat{A}+\hat{B}\hat{F},\ \hat{B},\ \hat{C}+\hat{D}\hat{F},\ \hat{D} \} \hat{R}^{-1/2} \qquad \text{(2-13a)}$$

is inner where $(\hat{R}^{1/2})^T(\hat{R}^{1/2}) = \hat{R} = \hat{D}^T\hat{D}$. $\hat{F}$ is given by

$$\hat{F} = -\hat{R}^{-1}(\hat{B}^T\hat{P}+\hat{D}^T\hat{C}) \qquad \text{(2-13b)}$$

and $\hat{P} \geq 0$ is the solution of the following algebraic Riccati equation [17]

$$(\hat{A}-\hat{B}\hat{R}^{-1}\hat{D}\hat{C})^T\hat{P} + \hat{P}(\hat{A}-\hat{B}\hat{R}^{-1}\hat{D}\hat{C}) - \hat{P}\hat{B}\hat{R}^{-1}\hat{B}^T\hat{P} + \hat{C}^T\hat{D}_{\perp}\hat{D}_{\perp}{}^T\hat{C} = 0 \qquad \text{(2-13c)}$$

where $\hat{D}_{\perp}$ is the orthogonal complement of $\hat{D}$. The notation $\hat{P} > 0$ ($\hat{P} \geq 0$) means that $\hat{P}$ is positive definite (nonnegative definite). The outer factor factor $\hat{S}_o(s)$ such that $\hat{S}(s) = \hat{S}_i(s)\hat{S}_o(s)$ is given by

$$\hat{S}_o(s) = \hat{R}^{1/2} \{ \hat{A}+\hat{B}\hat{F},\ \hat{B},\ \hat{F},\ I \}^{-1} = \hat{R}^{1/2} \{ \hat{A},\ -\hat{B},\ \hat{F},\ I \} \qquad \text{(2-13d)}$$

Theorem 2-2 : [4,12]

If $p > m$ in Theorem 2-1, then there exists a complementary inner factor $\hat{S}_{\perp}(s) \varepsilon (RH^{\infty})^{px(p-m)}$ such that the matrix $[\ \hat{S}_i \quad \hat{S}_{\perp}\ ]$ is square and inner. A particular realization is

$$\hat{S}_{\perp}(s) = \{ \hat{A}+\hat{B}\hat{F},\ -\hat{P}^{\dagger}\hat{C}^T\hat{D}_{\perp},\ \hat{C}+\hat{D}\hat{F},\ \hat{D}_{\perp} \} \qquad \text{(2-14)}$$

where $\hat{P}$ and $\hat{F}$ are from Theorem 2-1 and $\hat{P}^{\dagger}$ is the pseudo-inverse of $\hat{P}$.

In section III, we consider more general cases of the inner-outer factorizations for S(s) where the conditions $p \geq m$ and rank $D_{12} = m$ may not be satisfied.

The following theorem developed by Nett et al. [18] shows an easy way of computing the Bezout identity which will be used in section III.

Theorem 2-3: [18]

Given $G(s) = \{ A, B, C, D \} \varepsilon R(s)^{pxm}$. Then there exist $M_2(s) \varepsilon (RH^{\infty})^{pxp}$, $N_2(s) \varepsilon (RH^{\infty})^{pxm}$, $M_1(s) \varepsilon (RH^{\infty})^{mxm}$, $N_1(s) \varepsilon (RH^{\infty})^{pxm}$, $X_2(s) \varepsilon (RH^{\infty})^{pxp}$, $Y_2(s) \varepsilon (RH^{\infty})^{mxp}$, $X_1(s) \varepsilon (RH^{\infty})^{mxm}$, and $Y_1(s) \varepsilon (RH^{\infty})^{mxp}$ such that

$$M_2(s)^{-1}N_2(s) = G(s) \qquad \text{(2-15a)}$$

and

$$\begin{bmatrix} M_2(s) & N_2(s) \\ -Y_1(s) & X_1(s) \end{bmatrix} \begin{bmatrix} X_2(s) & -N_1(s) \\ Y_2(s) & M_1(s) \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \qquad \text{(2-15b)}$$

where

$$M_2(s) = \{ A+KC,\ K,\ C,\ I \} \qquad \text{(2-16a)}$$

$$N_2(s) = \{ A+KC,\ B+KD,\ C,\ D \} \qquad \text{(2-16b)}$$

$$X_2(s) = \{ A+BF,\ -K,\ C+DF,\ I \} \qquad \text{(2-16c)}$$

$$Y_2(s) = \{ A+BF,\ K,\ F,\ 0 \} \qquad \text{(2-16d)}$$

$$M_1(s) = \{ A+BF,\ B,\ F,\ I \} \qquad \text{(2-16e)}$$

$$N_1(s) = \{ A+BF,\ B,\ C+DF,\ D \} \qquad \text{(2-16f)}$$

$$X_1(s) = \{ A+KC,\ -(B+KD),\ F,\ I \} \qquad \text{(2-16g)}$$

$$Y_1(s) = \{ A+KC,\ K,\ F,\ 0 \} \qquad \text{(2-16h)}$$

Here F and K are chosen such that $A + BF$ and $A + KC$ are stable.

III. INNER-OUTER FACTORIZATIONS

There are two cases of inner-outer factorization problems arising in H^∞ optimization problems. They are problems (1-10) and (1-11) described in section I.

Case $p \geq m$:

First of all, we will consider problem (1-11), i.e., the case $p \geq m$. $S(s) \in (RH^\infty)^{pxm}$ is given by (2-8c), i.e.,

$$S(s) = \{ A+B_2F ,\ B_2 ,\ C_1+D_{12}F ,\ D_{12} \} \tag{3-1}$$

where

$$\{ A ,\ B_2 ,\ C_1 ,\ D_{12} \} = G_{12}(s) \tag{3-2}$$

and F is to be chosen such that $A+B_2F$ is stable. Numerically, F can be obtained from the following computations:

$$F = -R_2^{-1}B_2^TP \tag{3-3a}$$

and $P \geq 0$ is the solution of the following algebraic Riccati equation

$$A^TP+PA-PB_2R_2^{-1}B_2^TP+R_1 = 0 \tag{3-3b}$$

where $R_1 \geq 0$ and $R_2 > 0$ are arbitrary. Of course, F can be simplily chosen as 0 if A is stable.

Now, S(s) is thin or square, i.e., $p \geq m$. If rank $D_{12} = m$, i.e., $G_{12}(s)$ has no zeros at $s=\infty$, we can easily have a minimal realization $S(s) = \{ \hat{A}, \hat{B}, \hat{C}, \hat{D} \}$ with $\hat{D} = D_{12}$ of column rank and then Theorem 2-1 and 2-2 can be directly applied. That is, $S_i(s)$, $S_o(s)$, and $S_\perp(s)$ are given by (2-13a), (2-13d), and (2-14) respectively. A minimal realization of a given state-space model can be realiably computed by transforming [19,20] the model into a block Hessenberg form and deleting the uncontrollable and/or unobservable parts.

If rank $D_{12} < m$, i.e., $G_{12}(s)$ has zeros at $s=\infty$, the problem is more complicated. Theorem 2-1 and 2-2 cannot be applied directly. Our approach for solving this problem is to extract infinity zeros from S(s), i.e., to decompose S(s) into

$$S(s) = \hat{S}(s)\tilde{S}(s) \tag{3-4}$$

where $\tilde{S}(s) \in (RH^\infty)^{mxm}$ is outer and $\hat{S}(s) \in (RH^\infty)^{pxm}$ is of column rank at $s=\infty$. This decomposition can be easily made if S(s) can be represented by a ratio of a polynomial matrix over a polynomial function. Unfortunately, the conversion of a state-space description to a polynomial matrix description can introduce numerical instability. This difficulty can be avoided if $G_{12}(s)$ is given as

$$G_{12}(s) = N(s)/d(s) \tag{3-5}$$

where N(s) is a pxm polynomial matrix and d(s) is a polynomial function.

S(s) can be rewritten as

$$\begin{aligned} S(s) &= \{ A+B_2F ,\ B_2 ,\ C_1+D_{12}F ,\ D_{12} \} \\ &= \{ A ,\ B_2 ,\ C_1 ,\ D_{12} \} \{ A+B_2F ,\ B_2 ,\ F ,\ I \} = G_{12}(s)\, M(s) \end{aligned} \tag{3-6}$$

Here $G_{12}(s)$ is given by (3-2) and (3-5). If N(s) is not column reduced, we can find a unimodular matrix U(s) such that

$$N(s) = \tilde{N}(s)\, U(s) \tag{3-7}$$

and $\tilde{N}(s)$ is column reduced [21,22]. Let $n_i(s)$, $i=1,2,\dots,m$ be i-th column vector of $\tilde{N}(s)$. Decompose $n_i(s)/d(s)$ into

$$n_i(s)/d(s) = [\hat{n}_i(s)/\hat{d}_i(s)]\,[1/\tilde{d}_i(s)] \qquad i=1,2,\dots,m \tag{3-8}$$

such that $\deg \hat{n}_i(s) = \deg \hat{d}_i(s)$ and $\tilde{d}_i(s)$ has no unstable zeros. Then

$$G_{12}(s) = [\hat{N}(s)\hat{D}(s)^{-1}]\,[\tilde{D}(s)^{-1}U(s)] := \hat{G}_{12}(s)\, \tilde{G}_{12}(s) \tag{3-9}$$

where

$$\hat{N}(s) = [\ \hat{n}_1(s)\ \hat{n}_2(s)\ \ \ldots\ldots\ldots\ \ \hat{n}_m(s)\ ] \tag{3-10a}$$

$$\hat{D}(s) = \text{diag}[\ \hat{d}_1(s)\ \hat{d}_2(s)\ \ \ldots\ldots\ldots\ \ \hat{d}_m(s)\ ] \tag{3-10b}$$

$$\tilde{D}(s) = \text{diag}[\ \tilde{d}_1(s)\ \tilde{d}_2(s)\ \ \ldots\ldots\ldots\ \ \tilde{d}_m(s)\ ] \tag{3-10c}$$

It is easy to see that $\hat{G}_{12}(s) \epsilon R(s)^{pxm}$ is of column rank at $s=\infty$ and $\tilde{G}_{12}(s) \epsilon (RH^{\infty})^{mxm}$. Now

$$\begin{aligned} S(s) &= \hat{G}_{12}(s)\,\tilde{G}_{12}(s)M(s) \\ &= [\hat{G}_{12}(s)M(s)]\,[M(s)^{-1}\tilde{G}_{12}(s)M(s)] := \hat{S}(s)\,\tilde{S}(s) \end{aligned} \tag{3-11}$$

where

$$M(s) = \{A+B_2F\,,\ B_2\,,\ F\,,\ I\ \} \tag{3-12a}$$

$$M(s)^{-1} = \{A\,,\ -B_2\,,\ F\,,\ I\ \} \tag{3-12b}$$

$\tilde{S}(s) = M(s)^{-1}\tilde{G}_{12}(s)M(s) \epsilon (RH^{\infty})^{mxm}$ is outer and $\hat{S}(s) = \hat{G}_{12}(s)M(s) \epsilon (RH^{\infty})^{pxm}$ is of column rank at $s=\infty$. Let $\{\ \hat{A}, \hat{B}, \hat{C}, \hat{D}\ \}$ be a minimal realization of $\hat{S}(s)$. Then we can find the inner factor $\hat{S}_i(s)$, the outer factor $\hat{S}_o(s)$, and the complementary inner factor $\hat{S}_{\perp}(s)$ from (2-13a), (2-13d), and (2-14) respectively. Therefore, we have

$$S(s) = [\ \hat{S}_i(s)\ \ \hat{S}_{\perp}(s)\ ]\begin{bmatrix} S_o(s) \\ 0 \end{bmatrix} \tag{3-13}$$

where $S_o(s) = \hat{S}_o(s)\tilde{S}(s) \epsilon (RH^{\infty})^{mxm}$ is outer and $[\hat{S}_i(s)\ \hat{S}_{\perp}(s)] \epsilon (RH^{\infty})^{px(m+(p-m))}$ is inner.

Case $p < m$:

Next, we will consider problem (1-10), i.e., the case $p < m$. $S(s) \epsilon (RH^{\infty})^{pxm}$ is given by (2-8c), i.e.,

$$\begin{aligned} S(s) &= \{\ A+B_2F\,,\ B_2\,,\ C_1+D_{12}F\,,\ D_{12}\ \} \\ &= \{\ A\,,\ B_2\,,\ C_1\,,\ D_{12}\}\ \{\ A+B_2F\,,\ B_2\,,\ F\,,\ I\ \} = G_{12}(s)\,M(s) \end{aligned} \tag{3-14}$$

$G_{12}(s) \epsilon R(s)^{pxm}$ is given by

$$G_{12}(s) = N(s)\,/\,d(s) \tag{3-15}$$

where $N(s) \epsilon R[s]^{pxm}$ is fat, i.e., $p < m$ and $d(s)$ is a polynomial function. There exists a unimodular matrix $U_2(s)$ such that

$$N(s) = \tilde{N}(s)U_2(s) = [\ \tilde{N}_1(s)\ \ \tilde{N}_2(s)\ ]\,U_2(s) \tag{3-16}$$

with $\tilde{N}_1(s) \epsilon R[s]^{pxp}$ column reduced and $\tilde{N}_2(s) \epsilon R[s]^{px(m-p)}$. Let $n_i(s)$ be i-th column vector of $\tilde{N}(s)$. Decompose $n_i(s)/d(s)$ into

$$n_i(s)/d(s) = [\hat{n}_i(s)/\hat{d}_i(s)]\,[1/\tilde{d}_i(s)] \qquad i=1,2,\ldots,p \tag{3-17}$$

such that $\deg \hat{n}_i(s) = \deg \hat{d}_i(s)$ and $\tilde{d}_i(s)$ has no unstable zeros. Then we have

$$G_{12}(s) = [\ \hat{N}_1(s)\ \ \hat{N}_2(s)\ ]\begin{bmatrix} \hat{D}_1(s) & 0 \\ 0 & \hat{D}_2(s) \end{bmatrix}^{-1} \tilde{D}(s)^{-1}\,U_2(s) \tag{3-18}$$

where

$$\hat{N}_1(s) = [\ \hat{n}_1(s)\ \ \hat{n}_2(s)\ \ \ldots\ldots\ \ \hat{n}_p(s)\ ]\ \epsilon\ R[s]^{pxp} \tag{3-19a}$$

$$\hat{N}_2(s) = [\ n_{p+1}(s)\ \ \ldots\ldots\ \ n_m(s)\ ]\ \epsilon\ R[s]^{px(m-p)} \tag{3-19b}$$

$$\hat{D}_1(s) = \text{diag}\,\{\ \hat{d}_1(s)\ \ \hat{d}_2(s)\ \ \ldots\ldots\ \ \hat{d}_p(s)\ \}\ \epsilon\ R[s]^{pxp} \tag{3-19c}$$

$$\hat{D}_2(s) = d(s)\cdot I\ \ \epsilon\ R[s]^{(m-p)x(m-p)} \tag{3-19d}$$

$$\tilde{D}(s) = \text{diag}\,\{\ \tilde{d}_1(s)\ \ \tilde{d}_2(s)\ \ \ldots\ldots\ \ \tilde{d}_p(s)\ \ 1\ \ \ldots\ldots\ \ 1\ \}\ \epsilon\ R[s]^{mxm} \tag{3-19e}$$

Now

$$S(s) = [\,\hat{N}_1\hat{D}_1^{-1}\ \ \hat{N}_2\hat{D}_2^{-1}]\ \tilde{D}^{-1}U_2\,M$$

$$= [\,\hat{N}_1\hat{D}_1^{-1}\ \ \hat{N}_2\hat{D}_2^{-1}]\,M\cdot M^{-1}\,\tilde{D}^{-1}U_2\,M \tag{3-20}$$

Partition $M(s)$ as

$$M(s) = \begin{bmatrix} M_{11}(s) & M_{12}(s) \\ M_{21}(s) & M_{22}(s) \end{bmatrix} \in (RH^{\infty})^{(p+(m-p))x(p+(m-p))} \tag{3-21}$$

then

$$[\hat{N}_1\hat{D}_1^{-1}\ \ \hat{N}_2\hat{D}_2^{-1}]M = [\hat{N}_1\hat{D}_1^{-1}M_{11}+\hat{N}_2\hat{D}_2^{-1}M_{21}\ \ \ \hat{N}_1\hat{D}_1^{-1}M_{12}+\hat{N}_2\hat{D}_2^{-1}M_{22}]$$

$$:= [\,L_1(s)\ \ L_2(s)\,] \in (RH^{\infty})^{px(p+(m-p))} \tag{3-22}$$

Note that $L_1(s)$ is nonsingular at $s=\infty$. Let

$$L_1(s)^{-1}\,L_2(s) = \{\,\bar{A}, \bar{B}, \bar{C}, \bar{D}\,\} \in R(s)^{px(p-m)}$$

then from Theorem 2-3 we can find $M_2(s) \in (RH^{\infty})^{pxp}$, $N_2(s) \in (RH^{\infty})^{px(m-p)}$, $X_2(s) \in (RH^{\infty})^{pxp}$, $Y_2(s) \in (RH^{\infty})^{(m-p)xp}$, $M_1(s) \in (RH^{\infty})^{(m-p)x(m-p)}$, $N_1(s) \in (RH^{\infty})^{px(m-p)}$, $X_1(s) \in (RH^{\infty})^{(m-p)x(m-p)}$, and $Y_1(s) \in (RH^{\infty})^{(m-p)xp}$ such that

$$\begin{bmatrix} M_2(s) & N_2(s) \\ -Y_1(s) & X_1(s) \end{bmatrix}\begin{bmatrix} X_2(s) & -N_1(s) \\ Y_2(s) & M_1(s) \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{3-23a}$$

and

$$L_1(s)^{-1}L_2(s) = M_2(s)^{-1}N_2(s) = N_1(s)M_1(s)^{-1} \tag{3-23b}$$

From (3-23a) and (3-23b), we have

$$[\,L_1\ \ L_2\,] = [\,L_1\ \ L_2\,]\begin{bmatrix} X_2 & -N_1 \\ Y_2 & M_1 \end{bmatrix}\begin{bmatrix} M_2 & N_2 \\ -Y_1 & X_1 \end{bmatrix} = [\,\hat{S}\ \ \ 0\,]\begin{bmatrix} M_2 & N_2 \\ -Y_1 & X_1 \end{bmatrix}$$

$$= \hat{S}(s)\cdot[\,M_2(s)\ \ N_2(s)\,] = \hat{S}(s)\,.\,V_1(s) \tag{3-24}$$

where

$$\hat{S}(s) = L_1(s)X_2(s) + L_2(s)Y_2(s) \in (RH^{\infty})^{pxp} \tag{3-25a}$$

$$V_1(s) = [\,M_2(s)\ \ N_2(s)\,] \in (RH^{\infty})^{pxm} \tag{3-25b}$$

$\hat{S}(s)$ is nonsingular at $s=\infty$ since $X_2 = I$ and $Y_2 = 0$ at $s=\infty$. $V_1(s)$ is outer because its right inverse $[\,X_2(s)^T\ \ Y_2(s)^T\,]^T$ is in RH^{∞}.

Combine (3-20), (3-22), and (3-24), we have

$$S(s) = \hat{S}(s)V_1(s)M(s)^{-1}\,\tilde{D}(s)^{-1}U_2(s)M(s) := \hat{S}(s)\cdot\tilde{S}(s) \tag{3-26}$$

where $\hat{S}(s) \in (RH^{\infty})^{pxp}$ is nonsingular at $s=\infty$ and $\tilde{S}(s) = V_1M^{-1}\tilde{D}^{-1}U_2M \in (RH^{\infty})^{pxm}$ is outer. By using Theorem 2-1, we find an inner factor $\hat{S}_i(s)$ and an outer factor $\hat{S}_o(s)$ such that $\hat{S}(s) = \hat{S}_i(s)\hat{S}_o(s)$. Therefore we have the inner-outer factorization

$$S(s) = S_i(s)\,S_o(s) \tag{3-27}$$

where $S_i(s) = \hat{S}_i(s) \in (RH^{\infty})^{pxp}$ is inner and $S_o(s) = \hat{S}_o(s)\,\tilde{S}(s) \in (RH^{\infty})^{pxm}$ is outer.

IV. CONCLUDING REMARKS

In this paper, we have considered the inner-outer factorizations of $S(s) = \{\,A+B_2F,\ B_2,\ C_1+D_{12}F,\ D_{12}\,\} \in (RH^{\infty})^{pxm}$ for both cases $p \geq m$ and $p < m$ and D_{12} is not necessarily of column rank or of row rank. Our primary contribution is the introduction of the preliminary decomposition $S(s)$

$= \hat{S}(s)\tilde{S}(s)$ where $\tilde{S}(s)$ is outer and $\hat{S}(s)$ is of column rank at $s = \infty$. With this decomposition and Chu and Doyle's algorithm, all cases of inner-outer factorizations arising in multivariable H^∞ optimization problems can be computed.

ACKNOWLEDGMENT

The author would like to thank Professor J. B. Pearson and Professor T. L. Stewart for several useful discussions.

REFERENCES

[1] G. Zames, "Feedback and Optimal Sensitivity: Model Reference Transformations, Multiplicative Seminorms, and Approximate Inverses", IEEE Trans. A-C, AC-26, pp.301-320.

[2] B.A. Francis, J.W. Helton, and G. Zames, "H^∞-Optimal Feedback Controllers for Linear Multivariable Systems", IEEE Trans. A-C, AC-29, pp.888-900.

[3] B-C Chang and J.B. Pearson, "Optimal Disturbance Reduction in Linear Multivariable Systems", IEEE Trans. A-C, AC-29, pp.880-887.

[4] J. Doyle, Lectures Notes, ONR/Honeywell Workshop on Advances in Multivariable Control, Oct. 1984, Minneapolis, Minnesota.

[5] M.G. Safonov and M.S. Verma, "L^∞-sensitivity Optimization via Hankel-norm approximation", Proc. Amer. Cont. Conf., San Francisco, 1983.

[6] K. Glover, "All Optimal Hankel-norm approximations of Linear Multi- variable Systems and Their L^∞-error Bounds", Int. J. Control, Vol. 39, No. 6, pp.1115-1193.

[7] J. Doyle, "Synthesis of Robust Controllers and Filters", Proceedings of 22nd Conference on Decision and Control. Dec. 1983.

[8] B-C Chang and J.B. Pearson, "Iterative Computation of Minimal H^∞ Norm", To appear on Proceedings of 24th Conference on Decision and Control. Dec. 1985.

[9] C-C Chu and J. Doyle, "The General Distance Problem in H^∞ Synthesis", To appear on Proceedings of 24th Conference on Decision and Control. Dec. 1985.

[10] B-C Chang and J.B. Pearson, "Inner-Outer Factorization of Rational Matrices", Technical Report #8216, E.E. Dept., Rice University, 1982.

[11] C. Ganesh and J.B. Pearson, "Inner-Outer Factorization of Rational Matrices with a Complementary Inner Factor", Technical Report #8403, E.E. Dept., Rice University, 1984.

[12] C-C Chu and J. Doyle, "On Inner-Outer and Spectral Factorizations", Proceedings of 23rd Conference on Decision and Control. Dec. 1984.

[13] L. Cheng and J.B. Pearson, "Frequency Domain Synthesis of Multivariable Linear Regulators", IEEE Trans. A-C, AC-23, pp.3-15.

[14] L. Cheng and J.B. Pearson, "Synthesis of Linear Multivariable Regulators", IEEE Trans. A-C, AC-26, pp.194-202.

[15] D.C. Youla, J.J. Bongiorno, and H.A. Jabr, "Modern Wiener-Hopf Design of Optimal Controllers, Part II: The Multivariable Case", IEEE Trans. A-C, AC-21, pp. 319-338.

[16] C.A. Desoer, R.W. Liu, J. Murray, and R. Saeks, "Feedback System Design: The Fractional Representation Approach", IEEE Trans. A-C, AC-25, pp.399-412.

[17] A.J. Laub, "A Schur Method for Solving Algebraic Riccati Equations", IEEE Trans. A-C, AC-24, pp.913-921.

[18] C.N. Nett, C.A. Jacobson and M.J. Balas, "A Connection Between State-space and Doubly Coprime Fractional Representations", IEEE Trans. A-C, AC-29, pp.831-832.

[19] P. Van Dooren, "The Generalized Eigenstructure Problem in Linear System Theory", IEEE Trans. A-C, AC-26, pp.111-129.

[20] C-T Chen, Linear System Theory and Design, HRW, NY, 1984.

[21] W.A. Wolowich, Linear Multivariable Systems, Springer-Verlag, NY, 1974.

[22] T. Kailath, Linear Systems, Prentice-Hall, Englewood Cliffs, NJ, 1980.

NECESSARY AND SUFFICIENT CONDITION FOR STABILITY OF PERTURBED SYSTEM

QIU HUANG AND R. LIU
University of Notre Dame
Notre Dame, IN 46556

ABSTRACT

The problem of determining the stability of a multivariable closed-loop system in the presence of perturbations has attracted much attention. A considerable amount of the literature is devoted to the effect of perturbations on system stability and for system performance [1][2][3][4][5][6]. This, of course, is due to the fundamental importance of the issue. In order to discuss the robust stability, one important aspect is the characterization of the class of allowable perturbations for which a closed-loop system remains stable. In this paper, using algebraic theory of linear multivariable feedback systems, [7] we obtain a class of additive plant perturbations for which the perturbed closed-loop system remains stable.

PRELIMINAIRS

Throughout this paper, we assume the algebraic structure is the same as in [7].

Let G be a ring with identity and let H be a subring of G which includes identity. The feedback system and its subsystems will be represented by operators which are elements of G. The compensator will be such that the overall system will be represented by an operator in the subring H.

Define two multiplicative subsets of H

$$I = \{h \in H \mid h^{-1} \in G\} \tag{1}$$

i.e. I is the set of elements of H, which have an inverse in G;

$$J = \{h \in H \mid h^{-1} \in H\} \tag{2}$$

i.e. J is the set of unimodular in H.

Given the above algebraic structure we say that a system $P \in G$ has a right fractional representation in $\{G,H,I,J\}$, if there exist $N_r \in H$ and $D_r \in I$ such that $P = N_r D_r^{-1}$, furthermore, we say that the pair $(N_r,D_r) \in H\text{x}H$ is right coprime, if there exists U_r and V_r in H, such that

$$U_r N_r + V_r D_r = I \tag{3}$$

Similarily, the definition for left coprime of P is given, i.e. $P = D_\ell^{-1} N_\ell$ and there exist U_ℓ and V_ℓ such that

$$N_\ell V_\ell + D_\ell V_\ell = I \tag{4}$$

In otherwords, we assume that generalized Bezout identity exists [8],

$$\begin{bmatrix} N_r & V_r \\ D_\ell & N_\ell \end{bmatrix} \begin{bmatrix} U_r & -V_\ell \\ -D_r & U_\ell \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{5}$$

and

$$\begin{bmatrix} N_r & V_\ell \\ -D_r & U_\ell \end{bmatrix} \begin{bmatrix} U_r & -V_r \\ D_\ell & N_\ell \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix} \tag{6}$$

SYSTEM DESCRIPTIONS

Consider the standard feedback configuration shown in Fig. 1.

We have

$$\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} (I+PC)^{-1} & -P(I+CP)^{-1} \\ C(I+PC)^{-1} & (I+CP)^{-1} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{7}$$

Let's label the equation (7) as following:

$$e = H_{eu}(P,C)u \tag{8}$$

Definition 1: Assume that P and C have no unstable hidden mode, then the system shown in Fig. 1, denoted S(P,C), is stable, if $H_{eu}(P,C) \varepsilon H^{2x2}$, and $(I+PC)^{-1}$ exists.

Let the plant $P \varepsilon G$ be subjected to additive perturbation $\Delta P \varepsilon G$, thus becoming the additively perturbed plant $\tilde{P} = P + \Delta P$, and assume that $\tilde{P}$ has no unstable hidden mode, then the resulting additively perturbed feedback system, denoted $S(\tilde{P},C)$ shown in Fig. 2, is stable, if $H_{eu}(\tilde{P},C) \varepsilon H^{2x2}$, where

$$H_{eu}(\tilde{P},C) = \begin{bmatrix} (I+(P+\Delta P)C)^{-1} & -(P+\Delta P)(1+C(P+\Delta P))^{-1} \\ C(I+(P+\Delta P)C)^{-1} & (I+C(P+\Delta P))^{-1} \end{bmatrix} \tag{9}$$

and $[I+(P+\Delta P)C]^{-1}$ exists.

For simplicity of the formulas, we assume throughout the paper that $(I+PC)^{-1}$ exist.

Lemma 1

Let $\Delta H_{eu} = H_{eu}(\tilde{P},C) - H_{eu}(P,C)$. Assume S(P,C) is stable, then

$$S(\tilde{P},C) \text{ is stable } <=> \Delta H_{eu} \varepsilon H$$

Proof:
By hypothesis, $H_{eu}(P,C) \varepsilon H^{2x2}$. Using the closure properties of the ring H, and from the definition of ΔH_{eu}, we obtain that $S(\tilde{P},C)$ is stable, i.e. $H_{eu}(\tilde{P},C) \varepsilon H$, if and only if $\Delta H_{eu} \varepsilon H$, the claim is proved.

SYSTEM STABILITY

Now we consider in what a class of perturbations, the closed-loop system remains stable.

Lemma 2

Let $P = D_\ell^{-1}N_\ell = N_rD_r^{-1}$, (N_r,D_r), (N_ℓ,D_ℓ) is RCF and LCF of P, respectively in H, and $C = D_{c\ell}^{-1}N_{c\ell} = N_{cr}D_{cr}^{-1}$, (N_{cr},D_{cr}), $(N_{c\ell},D_{c\ell})$ is RCF and LCF of C, respectively in H. Under these assumptions, we have

$$\Delta H_{eu} = - \begin{bmatrix} D_{cr}R \\ N_{cr}R \end{bmatrix} M \, [\, LN_{c\ell} \quad LD_{c\ell} \,] \tag{10}$$

where

$$R = (D_\ell D_{cr} + N_\ell N_{cr})^{-1} \tag{11}$$

$$L = (D_{c\ell} D_r + N_{c\ell} N_r)^{-1} \tag{12}$$

$$M = D_\ell \Delta P[I + C(I + PC)^{-1} \Delta P]^{-1} D_r \tag{13}$$

Proof:

The proof is straitforward, although complex, and is therefore omitted.

Lemma 3

If R,L is as in lemma 2, and S(P,C) is stable, then

$(N_{cr}R,\ D_{cr}R)$ is right coprime,

and $(LN_{c\ell},\ LD_{c\ell})$ is left coprime, in H.

Proof:

Since S(P,C) is stable, this implies

$$(D_\ell D_{cr} + N_\ell N_{cr})^{-1} \ \varepsilon\ H,$$

i.e. R is unimodular, furthermore, (N_{cr}, D_{cr}) is right coprime, in H, then $(N_{cr}R, D_{cr}R)$ is right coprime [7], in H.

Similarily, $(LN_{c\ell}, LD_{c\ell})$ is left coprime in H.

Lemma 4

Let g ε G, and (x_r, y_r) be right coprime in H and (x_ℓ, y_ℓ) be left coprime in H then the following statement is equivalent:

I) $g \ \varepsilon\ H$

II) $x_r g x_\ell \ \varepsilon\ H$, $x_r g y_\ell \ \varepsilon\ H$

$y_r g x_\ell \ \varepsilon\ H$, $y_r g y_\ell \ \varepsilon\ H$

Proof:

I) → II) it is obvious, because of the closure properties of ring H.

II) → I) since (x_r, y_r) is right coprime in H, then there exist u_r, v_r, ε H, such that

$$u_r x_r + v_r y_r = I \tag{14}$$

postmultiply by gx_ℓ, in (14), we obtain

$$u_r x_r g x_\ell + v_r y_r g x_\ell = g x_\ell \tag{15}$$

the left handside of (15) belongs to H, which implies gx_ℓ ε H.

Similarily, gy_ℓ ε H.

Now, since (x_ℓ, y_ℓ) is left coprime in H, then there exist u_ℓ, v_ℓ ε H, such that

$$x_\ell u_\ell + y_\ell v_\ell = I \tag{16}$$

premultiply by g in (16), we obtain

$$g x_\ell u_\ell + g y_\ell v_\ell = g \tag{17}$$

the left handside of (17) belongs to H, which implies $g \in H$.

Theorem 1:

Consider the system $S(P,C)$ and $S(\tilde{P},C)$ defined in (7),(9). Assume $S(P,C)$ is stable, under these conditions the following statements are equivalent:

I) The perturbed system $S(\tilde{P},C)$ is stable,

II) $M \in H$, where M is defined in (13).

Proof:

Denote $\tilde{N}_{cr} = N_{cr}R$, $\tilde{D}_{cr} = D_{cr}R$, $\tilde{N}_{c\ell} = LN_{c\ell}$, $\tilde{D}_{c\ell} = LD_{c\ell}$. From lemma 1, we have $S(\tilde{P},C)$ is stable, if and only if $\Delta H_{eu} \in H^{2x2}$, so we only need to show that $\Delta H_{eu} \in H^{2x2}$, if and only if $M \in H$. Now, if $M \in H$, using the closure properties of the ring H, and from the fomular (10), we have $\Delta H_{eu} \in H^{2x2}$. Conversely, since $\Delta H_{eu} \in H^{2x2}$, from (10), we have

$$\tilde{D}_{cr} M \tilde{N}_{c\ell} \in H \, , \; \tilde{D}_{cr} M \tilde{D}_{c\ell} \in H$$

$$\tilde{N}_{cr} M \tilde{N}_{c\ell} \in H \, , \; \tilde{N}_{cr} M \tilde{D}_{c\ell} \in H.$$

Using Lemma 3,4, $M \in H$, the claim is proved.

Remark:

1) we have obtained an efficient formula for stability of additive perturbed system, there are no restrictions on the 'size' of the perturbations ΔP and the number of unstable poles of the perturbed plant P.

2) if we consider the more general system as shown in Fig. 3. Using the same description as in [10], under the assumptions that $S(P,C,F)$ is stable and $\tilde{P} = P + \Delta P$, then it can be shown that $S(\tilde{P},C,F)$, the perturbed system, is stable, if and only if

$$M = D_{p\ell}\Delta P[I + FC(I + PFC)^{-1}\Delta P]^{-1}D_{pr} \tag{18}$$

is stable, where $D_{p\ell}{}^{-1}N_{p\ell} = N_{pr}D_{pr}{}^{-1} = P$, (N_{pr},D_{pr}), $(N_{p\ell},D_{p\ell})$ is a RCF and LCF of P, respectively.

3) if we consider a interconnected system as shown in Fig. 4. Using the same description as in [11], and under the assumptions that $S(G)$ is stable, and $\tilde{G} = G + \Delta G$, then it can be shown that $S(\tilde{G})$, the perturbed system is stable, if and only if

$$M = D_{g\ell}\Delta G[I + F(I + GF)^{-1}\Delta G]^{-1}D_{gr} \tag{19}$$

is stable, where $G = N_{gr}D_{gr}{}^{-1} = D_{g\ell}{}^{-1}N_{g\ell}$, (N_{gr}, D_{gr}), $(N_{g\ell}, D_{g\ell})$ is a RCF and LCF of G, respectively, and $D_{gr} = \text{diag}\{D_{gr1},\dots D_{grk}\}$, $D_{g\ell} = \text{diag}\{D_{gi\ell},\dots D_{g\ell k}\}$, $F = \{F_{ij}\}$, $G = \text{diag}\{G_1,\dots G_k\}$

4) the significants of the theorem 1 can be illustrated by an example: Let

$$\tilde{P}(s) = \frac{100}{(s-1)(s+100)} \qquad (20)$$

A low-frequency model of the system is

$$P(s) = \frac{1}{s-1} \qquad (21)$$

Based on the low-frequency model, it can be shown that C = 2 can stabilize the feedback system $S(P,C)$. Based on Criterion (13), it can be shown that the perturbated system $S(\tilde{P},C)$ is also stable, i.e. it also 'stabilizes' the high frequency mode.

5) Using a different perturbation model, Bhaya and Desoer [9], has shown that the necessary and sufficient condition to stabilize a perturbed feedback system is given by

$$\Delta P[1+C(I+PC)^{-1}\Delta P]^{-1} \in H. \qquad (22)$$

It may be interesting to note that the above condition would show, in the above example, that the perturbed feedback system $S(\tilde{P},C)$ is unstable.

6) If denote 'y_3', shown in Fig. 2, as a 'pseudo' output, and need the maps $(u_1,u_2) \to (y_1,y_2,y_3)$ to be stable, instead of the maps $(u_1,u_2)| \to$ (y_1,y_2)(i.e.H_{yu}), then the Criterion (13) may be changed into

$$M = \Delta P[1+C(I+PC)^{-1}\Delta P]^{-1}D_r \in H. \qquad (23)$$

Based on this Criterion, in the above example, the perturbed feedback system $S(\tilde{P},C)$ is also stable.

In the following Corollary, we need the lemma 5 below:

Lemma 5

Let $w = x_\ell y^{-1} x_r$, where x_ℓ, $x_r \in H$, $y \in I$, and assume that (x_ℓ, y) is right coprime in H, (x_r, y) is left coprime in H. Under these conditions

$$w \in H \Leftrightarrow y^{-1} \in H \qquad (24)$$

Proof:

$\leftarrow$) it is obvious,

$\rightarrow$) if $w = x_\ell y^{-1} x_r \in H$, since (x_ℓ, y) is right coprime in H, then there exist u_r, v_r, $\in H$, such that

$$u_r x_1 + v_r y = I \qquad (25)$$

postmultiply by $y^{-1}x_r$, in (25), we have

$$u_r x_1 y^{-1} x_r + v_r x_r = y^{-1} x_r \qquad (26)$$

the left handside of (26) is in H, which implies $y^{-1}x_r$, in H.

Now, (x_r, y) is left coprime in H, then there exist u_ℓ, v_ℓ in H, such that

$$x_r u_1 + y v_1 = I \qquad (27)$$

Premultiply by y^{-1} in (27), we have

$$y^{-1}x_r u_1 + v_1 = y^{-1} \tag{28}$$

the left handside of (28) is in H, which implies $y^{-1} \varepsilon$ H.

Corollary 1:

If S(P,C) and ΔP is stable, then S(P,C) is stable, if and only if

$$[I + C(I+PC)^{-1}\Delta P]^{-1} \;\varepsilon\; H \tag{29}$$

Proof:

From theorem 1, we have $S(\tilde{P},C)$ is stable, if and only if $\Delta H_{eu} \;\varepsilon\; H^{2x2}$, if and only if

$$M = D_\ell \Delta P[I + C(I+PC)^{-1}\Delta P]^{-1}D_r \;\varepsilon\; H$$

Let $U_r,V_r;U_\ell,V_\ell$ satisfy (3),(4) and $C = V_r^{-1}U_r = U_\ell V_\ell^{-1}$, then $C(I + PC)^{-1} = D_rU_r \;\varepsilon\; H$ and $[I + C(I+PC)^{-1}\Delta P]^{-1} = (I + D_rU_r\Delta P)^{-1}$. Note that ΔP is stable, therefore

$$(D_r \;,\; I+D_rU_r\Delta P) \tag{30}$$

is left coprime in H. Using (6), we have $D_rU_r = U_\ell D_\ell$. Now $(D_\ell\Delta P,\; I + U_\ell D_\ell \Delta P)$ is right coprime in H, therefore

$$(D_\ell \Delta P \;,\; I + D_rU_r\Delta P) \tag{31}$$

is right coprime in H. From (30), (31), and lemma 5, we have M ε H, if and only if $(I + D_rU_r\Delta P)^{-1} \;\varepsilon\; H$. The corollary is proved.

Now we consider the robust stabilization of systems with uncertain parameters [13], for simplicity, we only discuss the perturbed system as $\tilde{P} = (I + K)P$, where P is the nominal plant. Assume that (I + K) ε J. Using the standard feedback configurations shown in Fig. 2

$$\Delta P = \tilde{P} - P = KP \tag{32}$$

Let $P = N_rD_r^{-1} = D_\ell^{-1}N_\ell$, (N_r,D_r), (N_ℓ,D_ℓ) is a RCF and LCF of P, respectively in H. $C = V_r^{-1}U_r$, (where $U_rN_r + V_rD_r = I$). Now we consider in what condition the perturbed feedback system remains stable, if S(P,C) is stable. Using theorem 1, the perturbed feedback system $S(\tilde{P},C)$ is stable, if and only if

$$M = D_\ell(I + KN_rU_r)^{-1}KN_r \tag{33}$$

is stable. By inspection, we have $(KN_r,\; I + KN_rU_r)$ is left coprime in H, $(D_\ell,\; I + KN_rU_r)$ is right coprime in H, because from (6), (U_r,D_ℓ) is right coprime in H, and

$$N_rU_r + V_\ell D_\ell = I$$

therefore, from (33), and lemma 5, we have M ε H, if and only if $(I + KN_rU_r)^{-1} \;\varepsilon\; H$. Hence we have proven Corollary 2 for the following multiplicative perturbation case.

Corollary 2:

Let P ε G, and ΔP ε G have the form

$$\Delta P = KP, \quad K + I \in J \tag{34}$$

and let S(P,C) be stable, then $S(\tilde{P},C)$ is stable if and only if

$$(I + KN_rU_r)^{-1} \in H \tag{35}$$

Remark:

1) The theorem can be applied to any linear system.

2) In the MIMO linear time-invariant case, lumped or distributed, it can be shown [12] that condition (35) is equivalent to

$$\det[I + KN_rU_r] \neq 0 \quad \forall S \in C_+ \tag{36}$$

which is equivalent to that the Nyquist plot of (36) does not go through the origin, nor encircle the origin.

3) When $K \in R_+ \setminus \{0\}$, i.e. positive real number, if K is very large, then from (36), the zeros of both plant P and the compensator C must be in $\mathring{C}_-$.

4) When $K \in R^{nxn}$, (36) gives an multiple gain-margin condition. When $K \in R(S)^{nxn}$, then (36) can be considered as a multiple phase-margin as well as multiple gain-margin condition.

PARAMETRIC REPRESENTATION OF THE STABLE SET OF ΔP

In this section, we first give a class of allowable perturbed plant, for which the perturbed system remains stable.

From Theorem 1, we have that the perturbed system $S(\tilde{P},C)$ is stable, if and only if

$$M = D_1\Delta P[I + C(I+PC)^{-1}\Delta P]^{-1}D_r \tag{37}$$

is stable, under the assumption that S(P,C) is stable. Let us now solve for ΔP from (37). Let

$$k = D_1\Delta PD_r \tag{38}$$

$$g = U_rD_1^{-1} \tag{39}$$

After a direct substitution, (37) becomes

$$M = k(I+gk)^{-1} \tag{40}$$

It can be shown from (40) that $k = M(I-gM)^{-1}$. From (38) and (39)

$$\Delta P = (D_1-MU_r)^{-1}MD_r^{-1} \tag{41}$$

We can state the following theorem:

Theorem 2

Assume that S(P,C) is stable, then $S(\tilde{P},C)$ is stable, if and only if

$$\Delta P \in \{\Delta P = (D_1 - WU_r)^{-1}WD_r^{-1} \mid W \in H, D_1 - WU_r \in I\} \tag{42}$$

Proof:

Let $S(\tilde{P},C)$ be stable then ΔP must saitsfy (41), with $M \varepsilon H$ and $(D_\ell - MU_r) \varepsilon I$. Therefore, (42) is satisfied with $W = M$. Conversely, let ΔP satisfy (42), for some W, we want to show that

$$M = k(I + gk)^{-1} \varepsilon H$$

where k and g are defined by (38) and (39). First,

$$(D_1 - WU_r) = (I - Wg)D_1 \varepsilon I$$

implies that $(I - Wg)^{-1}$ exists, and hence $(I - gW)^{-1}$ exists. It can be shown from (42) that

$$k = W(I + gk)$$

which together with the existence of the inverse $(I - gW)^{-1}$ implies that $(I + gk)^{-1}$ exists. Hence

$$W = k(I + gk)^{-1} = M \varepsilon H$$

by hypothesis. Therefore, $S(\tilde{P},C)$ is stable. The claim is proved.

Remark:

The set (42) is not empty, (because $W = 0, \check{P} = P$ belongs to the set (42)). And with some certain normed algebra for G, the set is an open, dense subset of G, which have both an RCF and LCF [14].

ACKNOWLEDGMENT

The authors wish to thank C. A. Desoer for his many constructive discussions. The first author also thanks Professor C. A. Desoer and Professor S. Sastry for teaching him advance control theory in his study in U. C. Berkeley. This research is supported in part by ONR Grant N00014-84K-0482.

REFERENCE

[1] Astrom, K. J., "Robustness of a design method based on assignment of poles and zeros," IEEE Trans. Autom. Control, Vol. AC-25, pp. 588-591, June 1980.

[2] Cruz, J. B., JR., Freudenberg, J. S., and Looze, D. P., "A relationship between sensitivity and stability of multivariable feedback systems," IEEE Trans. Autom. Control, Vol. AC-20, pp. 66-74, Feb. 1981.

[3] Desoer, C. A. and Wang, Y. T., "Foundations of feedback theory for nonlinear dynamical systems," IEEE Trans. Circuit and System, Vol. CAS-27, pp. 104-123, Feb. 1980.

[4] Doyle, J. C. and Stein, G., "Multivariable feedback design: concepts for a classical/modern synthesis," IEEE Trans. Autom. Control, Vol. AC-26, pp. 4-16, Feb. 1981.

[5] Safonov, M. G., Stability and Robustness of Multivariable Feedback Systems (Cambridge, MA, MIT Press), 1980.

[6] Desoer, C. A. and Liu, R. W., "Global parameterization of feedback systems with nonlinear plants," Systems and Control Letters, Vol. 1, No. 4, pp. 249-251, January 1982.

[7] Desoer, C. A., Liu, R. W., Murray, J. and Saeks, R., "Feedback system design: the fractional representation approach to analysis and synthesis," IEEE Trans. Autom. Control, Vol. AC-25, pp.. 399-412, 1980.
[8] Callier, F. M. and Desoer, C. A., Multivariable Feedback Systems, New York: Spring-Verlag, 1982.
[9] A. Bhaya and Desoer, C. A., "Robust Stability Under Additive Perturbations", Memorandum No. UCB/ERL M84/110 7 December 1984.
[10] M. J. Chen and C. A. Desoer, "Necessary and sifficient condition for robust stability of linear distributed feedback systems," Int. J. Contr., Vol. 35, pp. 255-267, Feb., 1982.
[11] M. J. Chen, and C. A. Desoer, "Algebraic theory for robust stability of interconnected systems: necessary and sufficient conditions," IEEE Trans. Autom. Contr., Vol. AC-29, No. 6, June, pp. 511-519, 1984.
[12] N. Jacobson, Basic Algebra I, San Francisco, CA, Freeman, 1974.
[13] Pramod P. Khargonekar and Allen, Trannenbaum, Proc. 23rd IEEE Conference on Decision and Control, December, pp. 1214-1217, 12-14, 1984.
[14] M. Vidyasager, H. Schneider and B. A. Francis, "Algebraic and Topological Aspects of Feedback Stabilization," IEEE Trans. Autom. Contr., Vol. AC-27, pp. 880-893, 1982.

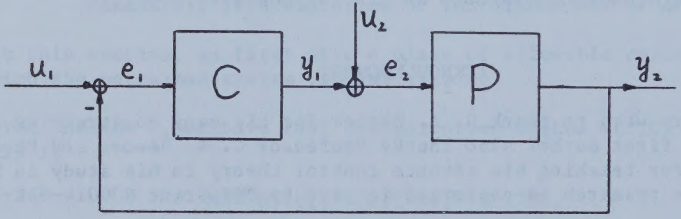


Figure 1

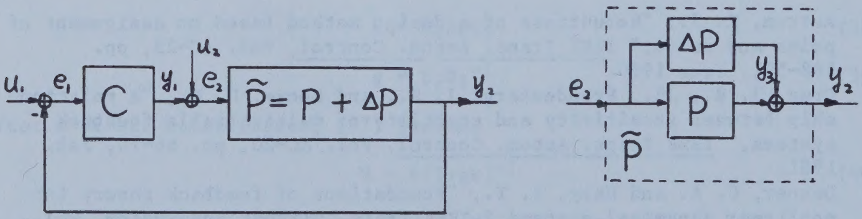


Figure 2

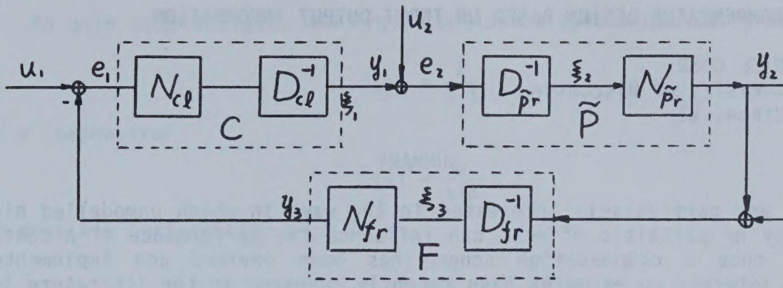

u₁
e₁
-
N_cl
D_cl^-1
ξ₁
C
y₁
u₂
e₂
D_p̃r^-1
ξ₂
N_p̃r
P̃
y₂
y₃
N_fr
ξ₃
D_fr^-1
F
u₃

Figure 3

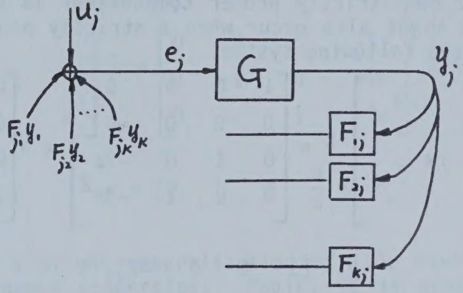

u_j
e_j
G
y_j
F_j1 y₁
F_j2 y₂
F_jK y_K
F_1j
F_2j
F_Kj

Figure 4

ROBUST COMPENSATOR DESIGN BASED ON INPUT-OUTPUT INFORMATION

DANIEL COBB
University of Wisconsin
Madison, WI

SUMMARY

We are particularly interested in the ways in which unmodelled high-frequency or parasitic effects can influence the performance of a control system, once a compensation scheme has been devised and implemented. Several interesting examples have recently appeared in the literature (see e.g. [1] and [2]), illustrating how parasitics can drive a closed-loop system to instability even when a conventional analysis of the best available model of the system predicts stability.

One important feature of [1] and [2] is that closed-loop instability occurs only when a non-strictly proper compensator is used. To see that these difficulties might also occur when a strictly proper compensator is applied, consider the following system:

$$\begin{bmatrix} 1 & & & \\ & \varepsilon & & \\ & & \varepsilon & \\ & & & \varepsilon^4 \end{bmatrix} \dot{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & -2\varepsilon \\ 0 & \varepsilon & 1 & -3\varepsilon^2 \end{bmatrix} x + \begin{bmatrix} 2 \\ 0 \\ 0 \\ 1 \end{bmatrix} u \tag{1}$$

$$y = [1 \quad 0 \quad 0 \quad -2]x$$

The system has eigenvalues equal to 1 and $-1/\varepsilon^2$, and as $\varepsilon \to 0^+$ the solutions of (1) converge to those of the $\varepsilon=0$ system in the sense of distributions and uniformly on compact subintervals of $(0, \infty)$, for every $x(0) \varepsilon \mathbf{R}^4$ and u in a broad class of admissible controls. The system (1) at $\varepsilon=0$ can be stabilized by applying the strictly proper compensator governed by

$$\dot{z} = -3z - 2y \tag{2}$$

$$u = z$$

However, for $\varepsilon > 0$ the characteristic polynomial of the closed-loop system is non-Hurwitz and in fact can be shown to have at least one root with real part tending to $+\infty$ as $\varepsilon \to 0^+$. Furthermore, it can also be proven that the solutions of the composite system determined by (1) and (2) diverge for certain inital conditions. It should be noted that the nature of the example (1) is somewhat different from those considered in [1] and [2] in that (1) contains a multi-rate perturbation.

Although the situation illustrated by (1) and (2) is worse than previously thought, we are convinced that there exist ways of designing around such problems [3]. Part of the reason that robust designs are still possible is that models such as (1) with $\varepsilon=0$ contain information concerning the internal system structure. A natural question to ask is to what extend dropping all but input-output information affects our ability to do robust compensation.

To gain some insight, one might consider a system nominally described by

$$P(s) = \frac{2}{s-1} \tag{3}$$

and a compensator

$$C(s) = \frac{\alpha}{s+\beta} \tag{4}$$

One possible perturbation of P is

$$P_\varepsilon(s) = \frac{2}{s-1} + \frac{4\varepsilon^2}{\alpha} \cdot \frac{s^2}{(\varepsilon^2 s+1)^3}$$

This perturbation is physically reasonable since it is realized by

$$\begin{bmatrix} 1 & & & \\ & \varepsilon & & \\ & & \varepsilon & \\ & & & \varepsilon^4 \end{bmatrix} \dot{x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & -2\varepsilon \\ 0 & \varepsilon & 1 & -3\varepsilon^2 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} u \tag{5}$$

$$y = [2 \quad 0 \quad 0 \quad \tfrac{4}{\alpha}]x$$

The system (5) is a slight generalization of (1), having the same eigenvalues and convergence properties. Again, it is easy to show that the characteristic polynomial of the closed-loop configuration is non-Hurwitz and has at least one root with real part tending to + ∞. Further, the corresponding solutions diverge for some initial conditions.

The important thing to noitce here is that a destabilizing perturbation of the form (5) exists for any first-order, strictly proper compensator (4). Hence, robust compensation cannot be achieved. We can prove a more general result:

Theorem. If the rational functions P(s) and C(s) are strictly proper and proper, respectively, there exists a parameterization $\{P_\varepsilon \mid \varepsilon \geq 0\}$ such that

1) the coefficients of P_ε converge to those of P as $\varepsilon \to 0^+$,

2) there is a singularly perturbed state-space realization of P_ε, each of whose eigenvalues is either fixed or equal to $-1/\varepsilon^p$ for some positive integer p and whose solutions converge in the sense of distributions and uniformly on compact subintervals of $(0, \infty)$ as $\varepsilon \to 0^+$ for every input and initial condition,

3) The closed-loop transfer function

$$H_\varepsilon = \frac{P_\varepsilon}{1-P_\varepsilon C}$$

has at least one pole λ_ε with $\operatorname{Re} \lambda_\varepsilon \to +\infty$,

4) the sequence $y_\varepsilon = Cx_\varepsilon$ does not converge in the distribution sense as $\varepsilon \to 0^+$.

If the class of perturbations described in 1) - 4) is taken to be physically realizable, the theorem states that robust compensation can never be achieved if only input-output information is available. More research is needed to demonstrate the realizability of this class of perturbations.

REFERENCES

[1] H. K. Khalil, "On the Robustness of Output Feedback Control Methods to Modelling Errors," IEEE Trans. Auto. Control, 26, April 1981.

[2] H. K. Khalil, "A Further Note on the Robustness of Output Feedback Control Methods to Modelling Errors," IEEE Trans. Auto. Control, 29, September 1984.

[3] D. Cobb, "Slow and Fast Stability in Singular Systems," 23rd IEEE Conf. on Decision and Control, 1984.

MATCHING CONDITIONS FOR OUTPUT FEEDBACK CONTROL OF UNCERTAIN LINEAR SYSTEMS

by

B. R. Barmish

Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, Wisconsin 53706

Abstract: This short paper provides new matching conditions which are useful in the stabilization of an uncertain linear system via output feedback.

I. INTRODUCTION

For the case of full state feedback, a number of stabilizing controls have been developed for uncertain systems satisfying so-called matching conditions; e.g., see [1] and [2] and their bibliographies. Reference [3] considers the case when the matching conditions are violated; it is shown that stabilizability can still be guaranteed provided that a certain measure of mismatch is sufficiently small. Quite recently, a new result on the problem of stabilization via output feedback has appeared; see [4]. For square systems satisfying a matching condition, it is shown that stabilization via output feedback is possible if the transfer function matrix associated with the nominal triple (A,B,C) is strictly positive real. Given (A,B,C), this positive realness requirement forces C to be in the range of B^T.

This paper is also concerned with stabilization via output feedback - - the main differences with [4] being as follows: We allow for nonsquare systems and require stabilizability of (A,B,C) instead of strict positive realness. Moreover, our stabilizability conditions involve the the "uncertainty structure" $(\Delta A,\Delta B,\Delta C)$. In the full version of this paper [5], it is shown that the results of [4] are a special case of those given here. In view of the existing literature, this short paper will have three focal points:

(i) We provide new matching conditions which are applicable to output feedback problems.

(ii) The new matching conditions, when specialized to the full state feedback case, describe a larger class of systems than those satisfying the existing matching conditions in [1] and [2].

(iii) We provide a "mismatch inequality" which is instrumental to the output feedback problem. For the special case of full state feedback, this new inequality degenerates into the mismatch inequality given in [3].

III. DESCRIPTION OF NEW MATCHING CONDITIONS

The System: Let $(A,B,C)\in R^{n\times n}\times R^{n\times m}\times R^{n\times r}$ be a stabilizable triple with rank B = m and rank C = r. Assume that $\Delta A(q)$, $\Delta B(q)$ and $\Delta C(q)$ are continuous matrix functions defined on a compact set $Q \subset R^p$ and let $A(q) \triangleq A + \Delta A(q)$, $B(q) \triangleq B + \Delta B(q)$ and $C(q) \triangleq C + \Delta C(q)$. We refer to $\{A(q),B(q),C(q),Q\}$ as the system (Σ).

Quadratic Stabilizability: The system (Σ) is said to be quadratically stabilizable (via output feedback) if there exists an $m \times r$ matrix K and an $n \times n$ positive-definite symmetric matrix P such that $[(A(q)+B(q)KC(q)),P] < 0$ where $[X,Y] \triangleq X^T Y + YX$ for two matrices X and Y. In this case, the pair (K,P) is said to achieve quadratic stabilization of (Σ).

New Matching Conditions: It is assumed that there exist continuous matrix functions $D_B(q)$, $D_C(q)$, $E_B(q)$ and $E_C(q)$ such that

$$\Delta A(q) = BD_B(q) + D_C(q)C;$$

$$\Delta B(q) = BE_B(q): \quad \|E_B(q)\| < 1;$$

$$\Delta C(q) = E_C(q)C; \quad \|E_C(q)\| < 1.$$

Stabilizing Controller and Mismatch Inequality: A control is constructed as follows: First select an $m \times r$ matrix F such that $A + BFC$ is stable. Next, choose any $Q = Q^T > 0$ and solve Lyapunov's equation $[A+BFC,P] = -Q$ for P. Given any $\gamma_1, \gamma_2 \in R$, define

$$M(\gamma_1,\gamma_2,q) \triangleq A(q) + B(q)FC(q) - \gamma_1 B(q)B^T P - \gamma_2 P^{-1}C^T C(q)$$

and select $\gamma_1, \gamma_2 \geq 0$ such that $[M(\gamma_1,\gamma_2,q),P] < 0$ for all $q \in Q$. (It is shown in [5] that appropriate γ_1, γ_2 always exist.) Finally, generate the output feedback gain matrix

$$K \triangleq F - \gamma_1 B^T P C^T (CC^T)^{-1} - \gamma_2 (B^T B)^{-1} B^T P^{-1} C^T.$$

In the full paper [5], it is show that (K,P) achieves quadratic stabilization of (Σ) if a certain mismatch inequality is satisfied: Define the "uncertain" projection matrices

$$\Psi_B(q) \triangleq B(q)(B^T B)^{-1} B^T - I; \quad \Psi_C(q) \triangleq C^T (CC^T)^{-1} C(q) - I$$

We now require that

$$\max_{q \in Q} \|\gamma_1 B(q)B^T P \Psi_C(q) + \gamma_2 \Psi_B(q) P^{-1} C^T C(q)\| < \lambda_{min}[Q]/2.$$

III. REFERENCES

[1] G. Leitmann, "Guaranteed Asymptotic Stability for Some Linear Systems with Bounded Uncertainties," Journal of Dynamic Systems, Measurement and Control, Vol. 101, no. 3, pp. 212-216, 1979.

[2] B. R. Barmish, M. Corless and G. Leitmann, "A New Class of Stabilizing Controllers for Uncertain Dynamical Systems," SIAM Journal of Control and Optimization, Vol. 21, no. 2, pp. 255-264, 1983.

[3] B. R. Barmish and G. Leitmann, "On Ultimate Boundedness Control of Uncertain Systems in the Absence of Matching Conditions," IEEE Transactions on Automatic Control, AC-27, no.1, pp. 153-158, 1982.

[4] A. Steinberg and M. Corless, "Output Feedback Stabilization of Uncertain Dynamical Systems," IEEE Transactions on Automatic Control, AC-30, no. 10, pp. 1025-1027, 1985.

[5] B. R. Barmish, "New Matching Conditions for Uncertain Linear Systems," in preparation.

PARALLEL INTERPOLATION SEARCH (EXTENDED ABSTRACT)

John H. Reif [*]
Harvard University
Cambridge, Mass. 02138

Dan E. Willard [†]
State University of New York at Albany
1400 Washington Avenue, Albany, New York
and Consultant, Bell Communications Research

Abstract

Several articles (see [2] thru [12]) have noted the usefullness of a retrieval algorithm called sequential interpolation search, and Yao and Yao have proven a lower bound log log N – O(1) showing this algorithm is actually optimal up to an additive constant on unindexed files of size N generated by the uniform probability distribution. We generalize the latter to show log log N – log log P – O(1) lower bounds the complexity of any retrieval algorithm with P parallel processors for searching an unindexed file of size N. This result is surprising because we also show how to obtain an upper bound that matches the lower bound up to an additive constant with a procedure that actually uses **no parallel processing** outside its last iteration (at which time our proposal turns on P processors in parallel). Our paper thus establishes the very surprising result that **parallel processing before the literally last iteration** in the search of an unindexed oredered file has **nearly no usefulness.** For the sake of simplicity, our theorems assume the file is generated by the uniform probability distribution.

1. Introduction

Searching an ordered file is a very common operation in data processing and so should be accomplished as rapidly as possible. Given N records, stored (ideally) in successive memory-locations in the order of their numeric keys $Y_1 < Y_2 \cdots < Y_N$, one often wishes to find a particular recod whose key equals y. For simplicity, the literature on interpolation search [2,3,7,8,9,12] generally assumes these N records were generated by the uniform distribution over the (0,1) interval and that this file is initially padded with the two pseudo-key boundary values of $Y_0 = 0$ and $Y_{N+1} = 1$. Each iteration i of interpolation search will search a subfile of the form $F_i = \{Y_{L_i} < Y_{L_i} < \cdots < Y_{R_i}\}$, where the contents of only the boundary keys Y_{L_i} ad Y_{R_i} is known at the start of the i-th iteration (for instance the first iteration has $L_1 = 0$, $R_1 = N+1$, $Y_{L_1} = 0$ and $Y_{R_1} = 1$). Interpolation search's i-th iteration will generate the cut address C_i^* below and check to see whether $Y_{C_i^*} = y$.

---

[*] Supported partially by National Science Foundation Grant NSF-MCS79-21024 and Office of Naval Research Contract N00014-80-C-0647.

[†] Supported partially by NSF Grant DCR-8412447.

$$C_i^* = L_i + \left\lceil \frac{(R_i - L_i - 1)(y - Y_{L_i})}{Y_{R_i} - Y_{L_i}} \right\rceil \qquad (1.1)$$

If $Y_{C_i^*} > y$ then the next iteration of interpolation search will search the subfile $F_{i+1} = \{Y_{L_i} < Y_{L_i+1} < \cdots Y_{C_i^*}\}$, and it will search the subfile $F_{i+1} = \{Y_{C_i^*} < Y_{C_i^*+1} < \cdots Y_{R_i}\}$ when $Y_{C_i^*} < y$. This process will continue until interpolation search either finds a record storing the key y or reduces the search problem to an empty state F_j (where $R_j = L_j+1$). Note interpolation search is identical to binary search except that it uses equation (1.1) rather than the formula $C_i = \lceil (L_i + R_i)/2 \rceil$ to define the address probed by the i-th iteration.

The desirable aspect of interpolation search is that it runs in expected time log log N when the set of keys are generated by the uniform distribution. Pearl et. al. [8,9] offer short proofs that interpolation search has complexity O(log log N), and Gonnet, Yao and Yao [2,12] provide a slightly more elaborate treatment that establishes the tighter upper bound of log log N + O(1). Yao and Yao [12] also prove interpolation search is optimal up to an additive constant by showing that no procedure can search unindexed ordered files in time better than log log N - O(1). Willard [10,11] illustrates modifications of interpolation search that are useful for probability distributions that are **both nonuniform** and **unknown** to the search algorithm, and [6] presents several useful generalizations of the latter result.

Throughout this paper, N_i will denote the number of unprobed keys in the subfile $F_i = \{Y_{L_i} < Y_{L_{i+1}} < \cdots Y_{R_i}\}$; in other words,

$$N_i = R_i - L_i - 1. \qquad (1.2)$$

Let P denote a fixed integer constant ≥ 2, and define VARIATION(P) to be an algorithm that is identical to interpolation search except that if the iteration i has $N_i \leq P$ then VARIATION(P) will turn on P parallel processors and reach a termination during this iteration. Our full-length paper shows VARIATION(P) searches files generated by the uniform probability distribution in expected time log log N - log log P + O(1) by generalizing the proof method from Yao and Yao. Intuition would suggest that even faster times would be possible if P parallel processors were used during each iteration of a search. However, the lower bound in Section 4 shows, to the contrary, that such further parallel processing improves performance by $\leq$ an additive constant. (By an additive constant, this paper means a quantity whose value is of course independent of both N and P.)

2. Intuition Behind Proof

The results announced in the last section may sound counter-intuitive, but the intuition behind their proof is actually quite simple. Let D_i denote the quantity defined below:

$$D_i = \mathrm{MIN}(R_i - C_i^*,\ C_i^* - L_i)$$

$$= \lceil \mathrm{MIN}(Y_{R_i} - y, y - Y_{L_i}) \cdot N_i \,/\, (Y_{R_i} - Y_{L_i}) \rceil \,. \tag{2.1}$$

A theme in the prior literature about sequential interpolation is that D_{i+1} satisfies the following equality with high probability

$$D_{i+1} \cong \sqrt{D_i}\,. \tag{2.2}$$

It is easy to verify that if equation (2.2) held with 100% certainty then log log D_1 iterations would be sufficient to reduce an initial file F_1 to a state j where $D_j \leq O(1)$. [2,12] establish that interpolation search runs in time log log D_1 + O(1) by showing equation (2.2) holds with extremely high probability as D_i approaches infinity. Further, [12] shows that no algorithm can run in time better than log log D_1 – O(1) by establishing essentially that no algorithms can cause the expected value of D_{i+1} to shrink faster than equation (2.2).

Our analogous analysis will indicate that any algorithm using P parallel processors satisfies equation (2.3) with high probability:

$$D_{i+1} \geq \Omega\,(\sqrt{D_i} \,/\, P). \tag{2.3}$$

A reader can appreciate part of the intuition behind our lower bound log log N – log log P – O(1) by noting that if equation (2.3) held with 100% probability and if $D_1 \cong N$ then clearly log log N – log log P – O(1) iterations would be necessary to reach a state where $D_j \leq O(1)$.

On the other hand, our full-length paper shows that log log N – log log P + O(1) iterations are on the average sufficient for interpolation search to produce a state where $N_j \leq P$ when the file is generated by the uniform probability distribution by generalizing Yao and Yao's proof method [12]. A consequence of this fact is that the expected number of iterations for VARIATION(P) to reach a termination is log log N – log log P + O(1). Hence, the combination of our upper and lower bounds show that parallel processing helps by no more than an additive constant when applied outside the very last iteration of interpolation search!

The next two sections prove these facts formally by examing the exact probabilities associated with our search process. As with prior literature about interpolation search, the formal analysis is more complicated than an intuitive overview, where the probability qualifications are absent.

One further point. The theorems in the next two sections technically discuss probabilistic bounds for the case of unsuccessful searches (that is, the case where the target key y is different from the N stored keys $Y_1 < Y_2 < \cdots < Y_N$). The same time bounds would hold up to an additive constant for successful searches, but it is preferable to discuss only unsuccessful searches for the sake of simplicity. Many of the earlier papers about sequential interpolation search make a similar simplification, where they discuss either successful searches or unsuccessful searches, but not both [2,12].

3. Background Information From Probability Theory

This section provides the machinery from probability theory for analyzing parallel interpolation search. Let B(j,N,p) denote the probability that N Bernoulli trials [1] produce j successes when each trial has a probability p of success. Then it is well known that

$$B(j,N,p) = \binom{N}{j}\, p^j\, (1-p)^{N-j}. \tag{3.1}$$

We now list three fairly easy consequences of classic probability that our paper will employ.

Proposition 3.1. $B(j,N,j/N) \leq O[\mathrm{MIN}(1; \frac{1}{\sqrt{j(N-j)}})]$

Proposition 3.2. $B(j,N,p) \leq O[\mathrm{MIN}(1;1/\sqrt{p(1-p)N})]$

Proposition 3.3. $\exists k > 0\ \forall p\ \forall N$ there is a probability $> k$ that at least $\lfloor pN \rfloor$ successes occur in a sequence of N Bernoulli trials where each individual trial produces a success with a probability $p > 0$.

Let $F_i = \{Y_{L_i} < Y_{L_{i+1}} < \cdots < Y_{R_i}\}$ once again denote the subfile that is examined in the i-th iteration of a search for the key y. Define the 5-tuple:

$$S_i = (L_i, R_i, Y_{L_i}, Y_{R_i}, y) \tag{3.2}$$

as the **state** of the i-th iteration of this search. Intuitively, this tuple represents all the information known about the file $F_i = \{Y_{L_i} < Y_{L_{i+1}} < \cdots < Y_{R_i}\}$ at the beginning of the i-th iteration of our search algorithms. Say X is a **state-dependent variable** iff it is a function of the 5-tuple S_i. Three examples of state-dependent variables are C_i^*, N_i and D_i (from equations (1.1), (1.2), and (2.1)). The variable D_i will play an especially important role in the discussion which follows.

Let Q_i^y denote the number of **interior** keys in the file $F_i = \{Y_{L_i} < \cdots < Y_{R_i}\}$ that satisfy $y_j < y$, and $\tilde{D}_i$ represent the quantity:

$$\tilde{D}_i = \mathrm{MIN}(Q_i^y,\ N_i - Q_i^y) \tag{3.3}$$

Say a variable is **file-dependent** iff its value is a function of the tuple $\{y, L_i, R_i, Y_{L_i}, Y_{L_i+1} \cdots Y_{R_i}\}$. Q_i^y and $\tilde{D}_i$ are examples of variables that are file-dependent but **not** state-dependent.

Assume X is a file-dependent variable. Let $\mathrm{Prob}(X \leq j \mid S_i)$ denote the probability that X is $\leq j$ assuming the interior keys in the file $F_i = \{Y_{L_i} < \cdots < Y_{R_i}\}$ are generated by the uniform distribution. Then this quantity is a state-dependent variable! Also X's expected value, denoted $E(X \mid S_i)$, is state-dependent.

The distinction between state and file-dependent variables will be greatly helpful in providing a simple but rigorous analysis for parallel interpolation search. A variable can be curiously state-dependent relative to one iteration but file dependent relative to another. For example, D_{i+1} is a state-dependent relative to S_{i+1}, but it is a file-dependent variable relative to S_i under all the search algorithms considered in this paper.

Lemma 3.4. The file-dependent variable Q_i^y satisfies the following three conditions:

A) For each $j \geq 0$ $\mathrm{Prop}(Q_i^y = j \mid S_i) = B(j, N_i, (y - Y_{L_i})/(Y_{R_i} - Y_{L_i}))$

B) If $y \leq (Y_{L_i} + Y_{R_i})/2$ then $E(Q_i^y \mid S_i) \leq D_i$

C) For each $j \geq 0$, $\mathrm{Prob}(Q_i^y = j \mid S_i) \leq O(1/\sqrt{D_i})$.

Lemma 3.5. The file-dependent variable $\tilde{D}_i$ satisfies the following two conditions:

A) $E(\tilde{D}_i \mid S_i) \leq D_i$.

B) $\mathrm{Prob}(\tilde{D}_i \leq 2D_i \mid S_i) \geq 1/2$.

Proof of Assertion A: Since $\tilde{D}_i \leq Q_i^y$, it follows that

$$E(\tilde{D}_i \mid S_i) \leq E(Q_i^y \mid S_i) \tag{3.5}$$

The combination of Lemma 3.4B and equation (3.5) imply $E(\tilde{D}_i \mid S_i) \leq D_i$ whenever $y \leq (Y_{L_i} + Y_{R_i})/2$. By a symmetry argument, the same result must be true when $y \geq (Y_{L_i} + Y_{R_i})/2$.

Proof of Assertion B: Since $\tilde{D}_i$ is always ≥ 0, Assertion B is an immediate consequence of Assertion A.

Q.E.D.

Lemma 3.6. There exists $k > 0$ such that every iteration i of interpolation search satisfies $\mathrm{Prob}(N_{i+1} \leq D_i \mid S_i) > k$.

Proof. With no loss of generality, we may assume $y \geq (Y_{L_i} + Y_{R_i})/2$. Then

$$\mathrm{Prob}(N_{i+1} \leq D_i \mid S_i) \geq \mathrm{Prob}(Q_i^y \geq N_i - D_i \mid S_i) \tag{3.6}$$

$$= \sum_{j=N_i-D_i}^{N_i} B(j, N_i, (y - Y_{L_i})/(Y_{R_i} - Y_{L_i}))$$

by Lemma 3.4A. Proposition 3.3 implies there exists some positive constant k, independent of N_i and D_i which bounds from below the latter term.

Q.E.D.

Define an **iterative search procedure with P parallel processes** as an algorithm which probes the subfile $F_i = \{Y_{L_i} < \cdots < Y_{R_i}\}$ by first generating P indices $C_i^1 \leq C_i^2 \leq \cdots \leq C_i^P$ and then checking for whether any of these indices satisfy $Y_{C_i^j} = y$. If the answer is yes then the parallel algorithm terminates; otherwise using a notation convention where $C_i^0 = L_i$ and $C_i^{P+1} = R_i$, the next iteration will search that subfile of the form $\{Y_{C_i^j} < \cdots < Y_{C_i^{j+1}}\}$ whose range contains y. As with interpolation search, this procedure continues until either the key y is found or the search space becomes empty, i.e. $N_i = 0$. (Note by equation (2.1), this condition also implies $D_i = 0$.)

Lemma 3.7. For each integer $m \geq 0$, every iterative search algorithm using P parallel processors satisfies $\mathrm{Prob}(D_{i+1} < m \mid S_i) \leq O(mP/\sqrt{D_i})$.

Proof. Lemma 3.4C implies each cut index C_i^j satisfies

$$\mathrm{Prob}(\ |\ C_i^j - Q_i^y - L_i\ | < m \mid S_i) \leq O(m/\sqrt{D_i}) \tag{3.7}$$

The definition of $\tilde{D}$ (i.e. equation (3.3)) implies

$$\tilde{D}_{i+1} = \mathrm{MIN}[\ |\ C_i^0 - Q_i^y - L_i\ |\ ,\ |\ C_i^1 - Q_i^y - L_i\ |\ ,\ \ldots,\ |\ C_i^{P+1} - Q_i^y - L_i\ |\ ]. \tag{3.8}$$

These two equations imply

$$\mathrm{Prob}(\tilde{D}_{i+1} < m \mid S_i) \leq O(mP/\sqrt{D_i}). \tag{3.9}$$

Equation (3.9) implies there exists some constant $k > 0$ such that

$$\mathrm{Prob}(\tilde{D}_{i+1} < m \mid S_i) \leq kmP/\sqrt{D_i}. \tag{3.10}$$

Applying Lemma 3.5B to the latter equation we conclude

$$\mathrm{Prob}(D_{i+1} < m/2 \mid S_i) \leq 2kmP/\sqrt{D_i}. \tag{3.11}$$

Since **all** real numbers $m > 0$ satisfy equation (3.11), so must **all** real numbres $m > 0$ satisfy

$$\mathrm{Prob}(D_{i+1} < m \mid S_i) \leq 4kmP/\sqrt{D_i} \leq O(mP/\sqrt{D_i}),$$

since 4k is a constant.

Q.E.D.

In the next section, log $\log^+ j$ denotes max(1,log log j), and LSUM(j) denotes the quantity:

$$\mathrm{LSUM}(j) = (1/j) \sum_{i=0}^{j-1} \log \log^+ i. \tag{3.12}$$

This definition implies:

Lemma 3.8. $\mathrm{LSUM}(j) \geq \log \log^+ j - O(1/\log j)$.

Our full-length paper combines Lemma 3.6 thru 3.8 with Yao and Yao's formalism to show that VARIATION(P) runs in expected time log log N – log log P + O(1). The remaining goal of the present paper is to show that no algorithm using P parallel processors can improve upon this result by more than an additive constant.

4. The Matching Lower Bound

Yao and Yao give a lower bound for the expected time of a sequential search into a random ordered table which we will now generaliza to any algorithm using P parallel processors.

Let SET(d) denote the set of all file states S_i satisfying $D_i \geq d$ and ALG(P) the set of all iterative search algorithms that use $\leq$P processors during each iteration. If $S \in$ SET(d) and $A \in$ ALG(P) then TIME(A,S) denotes the expected time that the algorithm A needs to process a state S when it is searching a uniformly generated file. Define $T_P(d)$ as the greatest lower bound on TIME(A,S) for the set of all ordered pairs (A,S) where $A \in$ ALG(P) and $S \in$ SET(d). Two immediate consequences of this definition are:

$$T_P(0) = 0 \tag{4.1}$$

$$\text{if } d_1 \leq d_2 \text{ then } T_P(d_1) \leq T_P(d_2). \tag{4.2}$$

Furthermore, the combination of equation (4.2) and Lemma 3.7 immediately imply

Lemma 4.1. There exists a constant $k > 1$ whose value is independent of both P and d that assures

$$T_P(d) \geq 1 + \frac{1}{\lceil \sqrt{d}/(kP) \rceil} \cdot \sum_{i=0}^{\lceil \sqrt{d}/(kP) \rceil - 1} T_P(i) \tag{4.3}$$

The chief goal of this section is to show recurrence relations (4.1) thru (4.3) imply $T_P(d) \geq \log\log d - \log\log P - O(1)$. Recall that LSUM was defined in equation (3.12). We state one preliminary lemma about this quantity before proving our main result.

Lemma 4.2. There exists constant $C > 0$ and $C^* > 0$ independent of both i and j, such that if $j \geq i^2 > 1$ then the following two equations hold:

$$\log\log(j/i) \geq \log\log(j) - C^* \log(i)/\log(j/i) \tag{4.4}$$

$$\text{LSUM}(\lceil j/i \rceil) \geq \log\log(j) - C \cdot \log(i)/\log(j/i). \tag{4.5}$$

Proof. It is straightforward to confirm equation (4.4), and equation (4.5) follows from the combination of equations (4.4) and Lemma 3.8.

Q.E.D.

Now we prove a proposition whose corollary is that $T_P(d) \geq \log\log d - \log\log P - O(1)$.

Theorem 4.3. Let $C > 0$ and $k > 1$ denote the constants defined in Lemmas 4.1 and 4.2. Then

$$T_P(d) \geq \log\log d - \log\log(kP) - 2 - C \cdot \text{MAX}\{0;2-2\log(kP)/\log d\}. \tag{4.6}$$

Proof by induction on d: It is obvious equation (4.6) is valid when $d \leq (kP)^4$, since the right side of (4.6) is then ≤ 0. For the case of $d > (kP)^4$, we may inductively assume equation (4.6) holds for all $d^* < d$. Then the combination of this assumption and equation (4.3) implies

$$T_P(d) \geq 1 + \text{LSUM}(\lceil \sqrt{d}/(kP) \rceil) - \log\log(kP) - 2 - \tag{4.7}$$

$$C \cdot \text{MAX}\{0;2-2\log(kP)/\log(\sqrt{d}/kP)\}$$

By substituting equation (4.5) and the facts $kP > 1$ and $d > (kP)^4$ into the inequality above, we get the desired conclusion that d also satisfies equation (4.6).

Q.E.D.

Corollary 4.4. $T_P(d) \geq \log\log d - \log\log P - O(1)$.

Proof. Since C and k were constants in Theorem 4.3 and since we always assume $P \geq 2$ when discussing parallel processing, it is evident that both $|\log\log(kP) - \log\log P|$ and $C \cdot \text{MAX}[0;2-2\log(kP)/\log d]$ are also bounded by constants independent of both P and d, i.e. they are O(1). >From those facts, it is evident that equation (4.6) implies Corollary 4.4.

Q.E.D.

We now state a second corollary which indicates that VARIATION(P) is optimal up to an additive constant.

Corollary 4.5. Suppose the records in the ordered file $F_1 = \{Y_1 < Y_2 \cdots Y_N\}$ are generated by the uniform distribution over (0,1) and this distribution also generates the search key y. Then $\log \log N - \log \log P - O(1)$ lower bounds the expected time for searching the file F_1 with any algorithm using $\leq P$ parallel processors.

Proof. Since y is generated by the uniform distribution, it is easily seen that $\text{Prob}(D_1 = i) = 2/N$, for each $i \leq N/2$. Therefore, by Corollary 4.4 and Lemma 3.8, the expected value of $T_P(D_1)$ is $\geq \log \log N - \log \log P - O(1)$.

Q.E.D.

References

1. Feller, W., *An Introduction to Probability Theory and Its Applications,* vol. I, Wiley, New York, third ed., 1968.

2. Gonnet, G.H., "Interpolation and Interpolation-Hash Searching" Ph.D. Thesis, Waterloo: University of Waterloo, 1977.

3. Gonnet, G.H., Rogers, L.D., and George, J.A., "An Algorithmic and Complexity Analysis of Interpolation Search," *Acta Informatica* 13, 39-52 (1980).

4. Knuth, D.E., *The Art of Computing Programming, vol. 3: Sorting and Searching,* Addison-Wesley, Reading, Mass., 1973.

5. Kruijer, H.S.M., "The Interpolated File Search Method," *Informatie* 16, 612-615 (1974).

6. Mehlhorn, K. and Tsakalidis, A., "Dynamic Interpolation Search," *Proceedings of ICALP-1985.*

7. Perl, Y., and Reingold, E.M., "Understanding and Complexity of Interpolation Search," *Infrm. Proc. Letters,* 6 (1977), 219-222.

8. Perl, Y., Itai, A., Avni, H., "Interpolation Search - A Log Log N Search," *Comm. ACM* 21, 550-557 (1978).

9. Peterson, W.W., "Addressing for Random-Access Storage," *IBM Journal of Research and Development* 1, 130-146 (1957).

10. Willard, D.E., "A log log N search algorithm for nonuniform distributions." Extended abstract in *Proceedings of the ORSA-TIMS Conference on the Applied Probability-Computer Science Interface,* vol. II, pp. 3-14, 1981; Full length version to appear in *SIAM J. on Computing,* Nov. 1985.

11. Willard, D.E., "Surprisingly Efficient Search Algorithms for Nonuniformly Generated Files," *21-st Allerton Conference on Communications, Control and Computing* (1983), pp. 656-662.

12. Yao, A.C., and F.F. Yao, "The Complexity of Searching an Ordered Random Table," *Proc. of the 17-th Annual Symposium on Foundations of Computer Science* (1976), 173-177.

SELF-ORGANIZATION AND ANNEALING

DANA RICHARDS
University of Virginia
Charlottesville, VA 22903

1. Introduction

A self-organizing data structure dynamically rearranges its records so that the expected access time is minimized. In the literature the linear list and the binary search tree are the most studied (e.g. [BITN79, RIVE76]). With a linear list each search proceeds sequentially from the first item and with trees all accesses begin at the root. In all cases an accessed record is moved "up", i.e. towards the beginning of the list or closer to the root. We reserve the term *self-organizing* for schemes which only use $O(1)$ extra storage; with enough extra storage truly optimal reorganization would be possible.

A linear list with one *finger*, an *f-list*, is a list of records with a global pointer to one of the records, called the finger. To access an item a sequential search from the current position of the finger is performed. The search is not a search for a key. (Otherwise, with an unsorted list, we would not know which direction, i.e. to the left or the right, to begin our search.) Instead we assume we know our destination, but use a sequential search due to the nature of the data structure. This formulation was motivated by the work of Carson and Reynolds [CARS85] on disk reorganization. There are two obvious advantages of f-lists over linear lists: the bidirectional reduces access time and, in many cases, it is better to conduct subsequent accesses in the vicinity of previous ones.

We are concerned with the problem of reducing access times for searchs on an f-list. Clearly by reorganizing the records we could reduce the expected seek time. There are two important differences from the list and tree problems. First, we assume that actually moving records is quite expensive and disruptive. Hence we would do our reorganization on a "virtual" f-list and after a period of time reorganize the entire f-list. In the previous work on lists and trees *expected access time* is a dynamic quantity, so the formulas do not necessarily hold for any particular snapshot of the data structure. We need a scheme which after a period of time will produce a good static arrangement. We have investigated using "annealing" techniques.

The second difference concerns the effect of dependent access probabilities. For lists and trees the importance of the fact that one record is likely to follow another record is greatly minimized since all accesses begin at the same location. (That is not to say dependent probabilities can be ignored [KONN81, LAM84].) Since a search on an f-list begins at the previous record's position the dependent probabilities would cause clustering in an optimal record arrangement. If the access probabilities are known to be independent then a simple symmetric "organ-pipe" arrangement is known to be optimal [WONG83]. However the general case is an NP-complete optimization problem [VAQU80]. (The reduction is from Optimal Linear Arrangement [GARE79].)

Simulated annealing is a technique for solving hard optimization problems that has recently found many applications [KIRK83]. Annealing is actually a framework for which detailed guidelines have yet to be found. Each new application involves much guesswork on how to set parameters, choose cost functions, update solutions, etc. The basic idea is that a trial solution is generated from a previous trial solution and the two are compared by a cost function. If the new solution has lower cost it is accepted, otherwise it is accepted probabilistically according to the current "temperature" (usually with respect to the Boltzmann distribution.) As this process iterates the temperature decreases leading to a "frozen" locally optimal solution.

Our application of simulated annealing is somewhat different from those in the literature. Since we allow only $O(1)$ extra storage it is impossible to have a global cost function that depends on the arrangement of more than $O(1)$ records. Given that we must make a blind choice we considered two alternatives. First we could have each solution cost no more than the previous solution, that is we always accept it. Second we could assume the new solution costs more by an amount which is a function of the most recent access. In particular, the reorganization implied by accessing a record is accepted probabilistically as a function of the access time and temperature. The first alternative is a degenerate case of the second.

We began our investigation by looking at the simpler (and presumably related) problem of linear lists without fingers. We used several self-organization techniques with several particular annealing procedures to produce final static orderings of a set of records. Given the (independent) access probabilities it is well-known that the optimal static ordering is in decreasing order of access probability. Since these were known to us we could easily evaluate each static ordering produced by our algorithm by computing the expected access time and calculating the percentage over the optimal ordering.

We have performed hundreds of trials to search for the best self-organization technique, the best annealing schedule, and the best cost function. (We feel that there are many more schedules and functions to be tried.) We did half the trials with a linear distribution and the rest with Zipf's distribution. The latter better models the (stationary) access probabilities for disks and file accesses. The most surprising aspect of our work is the robustness of the naive method which has a constant (0) cost function, i.e. unaffected self-organization. Further, it rarely made even a negligible improvement to vary the cost function though often it would produce much worse solutions, presumably due to the effect of slow convergence rates. The Zipf distributed data produced similar results to the linear data, only with even slower convergence.

After the experiments with linear lists we did many trials with f-lists. Instead of using random data we used actual access sequences from normally operating disks. Here it is much harder to evaluate our results since we cannot determine the optimal arrangement. We measured, with an actual sequence of accesses, the average search time of the initial ordering and the final static ordering. The self-organization techniques were the same as before only we moved toward the previous record instead of the front. Again we found the naive algorithm to be incredibly robust.

2. Self-Organizing Linear Lists

In this section we review what is known about self-organizing linear lists. We restrict our attention in this paper to rules that move an accessed record towards the beginning of the list and leaves the remaining records in the same relative order. Let p_i be the independent probability of accessing R_i, the ith record. Let C_A be the cost or average access time in the limit if the rule A is used for self-organization. OPT is the trivial rule that maintains the list in static order by decreasing access probabilities. OPT is used for comparison since it is the optimal static ordering [MCCA65]. Let there be n records, $R_0, \ldots, R_{n-1}$, and m accesses, where $m \to \infty$ unless otherwise stated.

The best known techniques are *transpose* (TR), moving one step towards the front, and *move-to-front* (MTF), moving all the way to the front. It is known that

$$C_{\mathrm{MTF}} = 1 + 2 \sum_{i<j} \frac{p_i p_j}{p_i + p_j}$$

and that $C_{\mathrm{MTF}} \leqslant \frac{\pi}{2} C_{\mathrm{OPT}}$ [CHUN85]. A complicated formula for the probability that R_i is in position j is known [BURV73]. The most important result about TR is that $C_{\mathrm{TR}} \leqslant C_{\mathrm{MTF}}$ [HEND76, RIVE76]. Let $b(i,j)$ be the probability in the limit that that R_i is before R_j. The reason analytic results are easy to obtain for MTF but not for TR (or for f-lists as well) is that in the latter case $b(i,j)$ is hard to calculate directly. MTF has been analyzed for several distributions [GONN81] including Zipf's distribution for which $C_{\mathrm{MTF}} \leqslant 2\ln 2\, C_{\mathrm{OPT}} = 1.38 \cdots C_{\mathrm{OPT}}$.

While TR is analytically better than MTF, MTF is often preferable in the short run due to its rapid convergence (e.g. see [TENE78, YU79] for empirical evidence). From a combination of analytic and empirical results a rule of thumb is to prefer TR over MTF if the number of accesses is $\Omega(n^2)$ [BITN79]. There are two approaches to the problem of the relationship of convergence to the number of accesses. First a hybrid rule [BITN79] which switches from MTF to TR after an appropriate number of accesses has been proposed. Second a "spectrum" of rules, with TR and MTF at the extremes, can be selected from depending on the number of accesses. In section 4 we smoothly combine these approaches.

The first spectrum proposed was MF_k, "move forward k" positions or to the front whichever is closer [RIVE76]. We have $C_{\mathrm{MF}_{k-1}} \leqslant C_{\mathrm{MF}_k}$ which implies $C_{\mathrm{TR}} \leqslant C_{\mathrm{MTF}}$ [GONN81]. But empirical results show that as n/m increases then k should increase as well, as expected due to the convergence rates [TENE78]. Other spectra divide the list of records at some point and use different rules, MF_{k_1} and MF_{k_2} for records before and after the dividing point (which acts like a "front" for the second half) respectively [TENE78, TENE82]. All analytic and empirical evidence indicates that we should have $k_1 \leqslant k_2$ and that the more "TR-like" a rule is the better the asymptotic cost. Of course this generalizes to many division points [TENE78] leading to the following elegant rule [SLEA85a]. MPF_k, "move proportionately forward", is the rule that moves the record in position *pos* forward $\lfloor pos/k \rfloor$ positions. Intuitively the advantage of this rule is that it allows records near the end of the list to move rapidly but those near the front move more slowly leading to the stability that makes TR asymptotically better. MPF_k is the basic rule we used in our experiments.

In this paper we deal with time varying rules where it becomes increasingly unlikely a move will be acted on after an access. In a similar vein the "k-in-a-row" rule was proposed [GONN81, KAN80] where a record is moved, by whatever rule, only if accessed k consecutive times. As k increases the rule becomes "more static". Let TR_k and MTF_k stand for the corresponding rules. It is known that $C_{\mathrm{TR}_k} < C_{\mathrm{MTF}_k}$ and $C_{\mathrm{TR}_k} < C_{\mathrm{TR}_{k-1}}$. Let the records be numbered so that $p_0 > p_1 > \cdots > p_{n-1}$, assuming distinct values for now. A rule is *progressively reluctant*, for parameter k, if

$$p_i^{(k)} / \sum_{j=i}^{n} p_j^{(k)} \to 1 \quad as \;\; k \to \infty, \; 1 \leqslant i \leqslant n,$$

where $p_i^{(k)}$ is the probability that R_i will be moved. For the class of rules in this paper every k-in-a-row rule is progressively reluctant and it can be shown that as $k \to \infty$ the probability that R_i is in position i goes to 1, i.e. the same as OPT [KAN80]. A more reluctant variant of k-in-a-row is the "k-in-a-batch" rule which requires that the k-in-a-row occurs in predetermined intervals [GONN81]. Let $\mathrm{MTF}_k{}'$ be the corresponding rule. It is known that $C_{\mathrm{MTF}_k{}'} \leqslant (\frac{\pi}{2k} \csc \frac{\pi}{2k}) C_{\mathrm{OPT}}$ and that for $k=2$ and 3 the constant is $1.110\cdots$ and $1.04\cdots$ respectively [CHUN85].

There are more direct ways to process fewer accesses. For independent access probabilities it does not matter if accesses are only periodically used for reorganization [MCCA65]. In fact if a rule is ignored with probability α, α fixed, the asymptotic results are unaffected [KAN80]. However consider the case of ignoring an access at position i with probability $\alpha_i > 0$. The analyses of TR are unaltered since the underlying Markov process is time-reversible [KAN80]. We conjecture that the expected access time is never worse for any rule in this paper if $\alpha_0 < \cdots < \alpha_{n-1}$.

When we assume the access probabilities are dependent it is typical to assume the accesses are a Markov process themselves, i.e. p_{ij}, the probability of accessing R_j after R_i, is independent of previous accesses. Let m_{ij} be the *mean first passage time* from R_i to R_j. Recall for the independent case $m_{ij} = 1/p_j$. It is known [LAM84] that

$$C_{\mathrm{MTF}} = 1 + 2 \sum_{i<j} \frac{1}{m_{ij} + m_{ji}}$$

and

$$C_{\mathrm{MTF}} \leqslant (1-2a) + 2aC_{\mathrm{OPT}}$$

where $a = 1 - \min_i\{p_{ii}\}$. Note that we can have $C_{\mathrm{MTF}} < C_{\mathrm{OPT}}$. Further there are distributions for which $C_{\mathrm{MTF}} < C_{\mathrm{TR}} < C_{\mathrm{OPT}}$. (There are distributions for which TR and MTF perform worse than a random organization, as well.) The intuition is that in highly "coupled" distributions MTF should perform well since it rapidly brings together a cluster of records that TR would be slow to respond to and that OPT, being static, would not respond to at all. Further, several authors have noted that if the pattern of access is itself time varying then OPT could be worse than a dynamic rule, even for the independent case.

To complete this survey of results we should mention that for the class of rules in this paper it has been long conjectured that TR is the optimal rule for independent accesses, if indeed there is a unique optimal rule. There has been much work on this [GONN81, KAN80, RIVE76]. However it is known that if when accessing position i an arbitrary permutation, depending on i, can be applied to the list of records, then TR is not always optimal [ANDE82]. If we generalize the class further to allow arbitrary extra storage for frequency counts then rules with costs equal to C_{OPT} are trivial, e.g. a list kept in decreasing frquency order (see also [BITN79, LAM81]). If we are concerned with the *amortized cost*, the sum of the access times for m accesses, then it is known that MTF performs very well in the worst case and TR does not [BENT85, SLEA85a]. However these considerations are mainly insurance against wildly improbable access sequences.

Oommen has done some preliminary work on using self-organization to produce static list orderings [OOMM84]. However his scheme uses $O(n)$ extra storage for frequency counters which are used with a probabilistic MTF rule. Not only is the rule complicated, e.g. using periodic sorting steps, but is inferior to the trivial rule mentioned above. He mentions, in a final note, that a probabilistic "move-to-rear" rule is far simpler, but still it uses $O(n)$ extra storage.

3. Simulated Annealing Schedules

In this section we present our annealing algorithm which shares many details with other such algorithms in the literature but is tailored to our application. The procedure begins with a temperature t which drops over time. It does not drop "continuously" but in discrete steps; at each step a certain amount of activity must occur.

The algorithm is in figure 1. Each access is considered for action by some reorganization rule *moverule*. We use $\Delta(access, t)$ to evaluate the change in "cost" (not necessarily the average access time) between the current configuration of the list and the list resulting from *moverule*$(access, t)$. If $\Delta < 0$ we accept the move, otherwise we accept probabilistically according to the Boltzmann distribution. Clearly as $t \to 0$ the list "freezes" since no moves will be permitted. However to anticipate this and save computation time we use the counters *att* and *succ*. If far more attempts than successes occur in some consecutive time period then the list is judged to be frozen. One stage of the algorithm tries much reorganization at a fixed temperature. Depending on the number of successes the stage is judged as being a "failure". After each stage the temperature is decremented by the factor δ, $\delta<1$. The temperature must remain above an arbitrary *limit* to save computation time.

Filling in the details of this algorithm was done mostly by trial and error. We had $fbound=3$, $abound=50n$, $sbound=10n$, $t_0=100$, and $limit = t_0/100$. Typically $\delta=.9$ but varying δ was useful for determining convergence rates.

Actually we never have $\Delta<0$ so that test is vacuous. It was included to emphasize the difference of our algorithm from previous schemes. No attempt was made at evaluating the "cost" of the lists, before and after a move, due to the $O(1)$ extra storage restriction. Instead we assumed that $\Delta \geqslant 0$ and it grew monotonically with the search time for the accessed item. This heuristic assumes that some degree of organization is present and outlying records are less important. This assumption is clearly time varying, hence the second parameter to Δ.

```
$t \leftarrow t_0$
$frozen \leftarrow false$
$failures \leftarrow 0$
while not $frozen$ and $t > limit$ do
    $att \leftarrow succ \leftarrow 0$
    while $att < abound$ and $succ < sbound$ do
        get($access$)
        $\Delta \leftarrow \Delta(access, t)$
        if $\Delta < 0$ or $random < e^{-\Delta/t}$ then
            $moverule(access, t)$
            $succ \leftarrow succ + 1$
        endif
        $att \leftarrow att + 1$
    endwhile
    if $att \geqslant abound$ then
        $failures \leftarrow failures + 1$
    else
        $failures \leftarrow 0$
    endif
    $frozen \leftarrow failures = fbound$
    $t \leftarrow \delta t$
endwhile
```

Annealing algorithm

Figure 1

4. Annealing Linear Lists

Our investigation began with linear lists before considering f-lists. No attempt was made at an exhaustive multivariate analysis; instead we sought to identify those parameters and other details that gave improved performance. Our statistics are the average of only 5 runs. The effect of the variance was evident but did not affect the relative rankings of the schemes. No attempt was made at determining higher moments. Our measure of performance was the percentage of C_{OPT}.

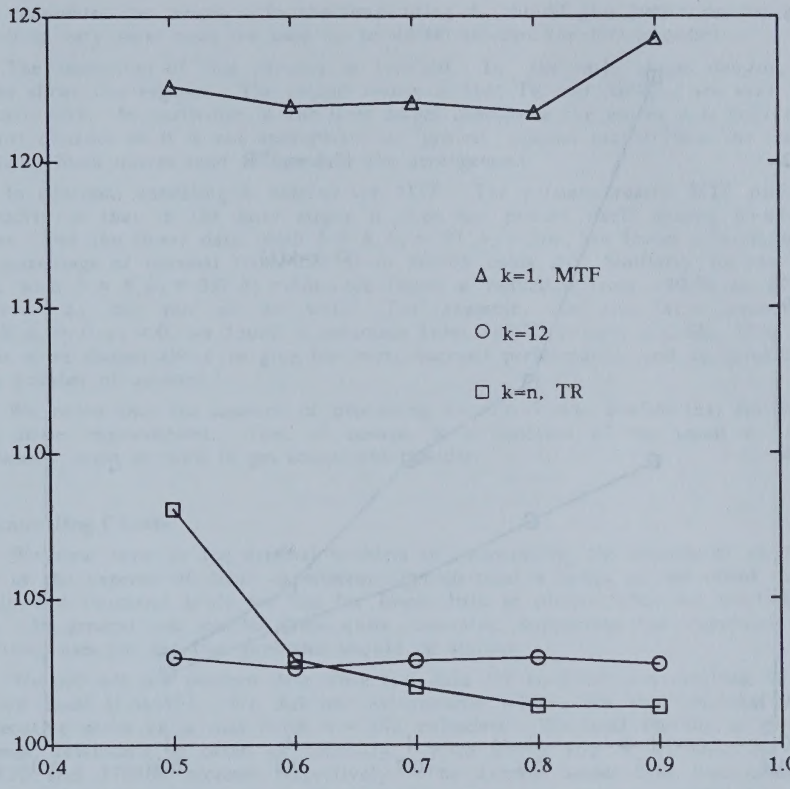
We used random access sequences for two different distributions, with $n = 100$ throughout. We generated an access randomly with respect to "frequency" counts. The *linear* distribution used random frequencies from 0 to 9999. The *Zipf* distribution used frequencies equal to $\lfloor 1000/i \rfloor$, $1 \leqslant i \leqslant 100$, which were then randomly permuted. (This closely approximates the actual Zipf's distribution, $p_i = 1/iH_n$.) It is easy to predict that the linear data would be less sensitive than the Zipf data. This is because the latter contains relatively few large values, all of which must be very close to the front of the list.

Our rule, *moverule*, was to move forward a distance $\lfloor pos/k \rfloor$ when accessing a record in position pos, i.e. MPF_k. With $k = 1$ and n we have MTF and TR respectively. But, as discussed in section 2, it is better to have a time varying rule. Recall that the temperature can vary from t_0 to $limit$ and at the ith stage $t = t_0 \delta^i$. Let

$$\sigma(t) = 1 - \frac{t - limit}{t_0}.$$

Note that $\sigma(t)$ varies from $limit/t_0$ to 1 as a function of the temperature. We chose $k = k(t) = \lfloor \sigma(t) n \rfloor$. If $limit \leqslant t_0/n$ then this rule varies smoothly from MTF to TR, with the majority of the time it being "TR-like".

Let us begin by having $\Delta = 0$, i.e. all accesses are acted on. We effectively increase the length of the experiment by increasing δ. We first considered fixed k's, $k = 1$, 12, and 100, as shown in figure 2 for linear data. We see that for MTF, $k = 1$, the performance is relatively insensitive to δ showing that it is near its steady



δ vs. % of optimal

Linear data with $\Delta = 0$

Figure 2

state. Further, while its performance is well within its theoretical limits it is far worse than TR, $k=n$, indicating that the convergence "crossover" of the two methods is long past. Such a crossover is seen for MPF_{12} and TR. Similar results were obtained for the Zipf data. For reference, the number of accesses for δ = .5, .6, .7, .8, and .9 were 7000, 10000, 13000, 21000, and 44000 respectively.

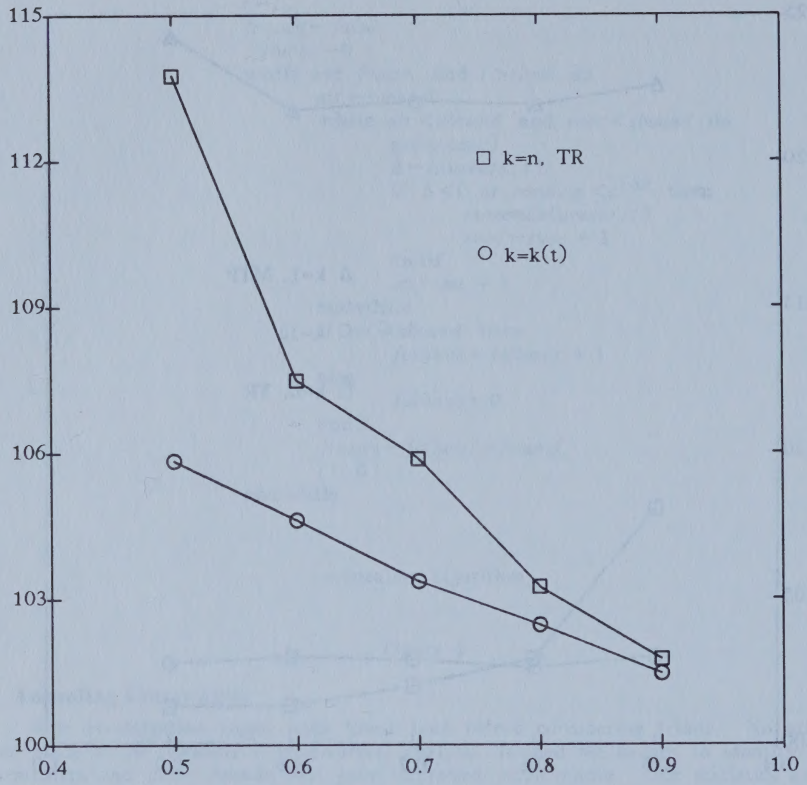
We compared the time varying $\mathrm{MPF}_{k(t)}$ with TR, $k=n$, again with $\Delta = 0$. The results are shown in figure 3 for the Zipf data. The results indicate a clear cut improvement using $k(t)$. Recall that the same records are being moved in both cases, only the distance moved is different. A similar improvement was found for the linear data.

The next step is to introduce $\Delta \neq 0$ so that annealing occurs. Recall an access to a record *access* in position *pos* has been made, so we let $\Delta(access, t) = \Delta(pos, t)$. We tried several functions, notably

$$\Delta_1(pos, t) = a_1\, pos + a_2$$

and

$$\Delta_2(pos, t) = b_1((t_0/t)^{b_2\, pos} - 1)$$



δ vs. % of optimal

Zipf data with Δ = 0

Figure 3

where a_1, a_2, b_1, and b_2 are constants which may depend on n. The motivation for Δ_1 is simply that, in steady state, elements further from the front are less likely to need to go towards the front. It has the disadvantage of being independent of t and hence is applied before the steady state and, in fact, should delay convergence.

There are simpler functions that depend on *pos* and t than Δ_2. For example $c_1\, pos/t + c_2$. However Δ_2 is the simplest intuitively motivated function that gives rise to a rule that is progressively reluctant. As stated in section 2, this ensures that the scheme is probabilistically optimal as $t \to 0$. Note that Δ_2 is initially 0 and grows with both t and *pos*. Hence in the beginning most accesses are acted on which should speed convergence. It is important that Δ not grow so fast that the annealing schedule "freezes" before enough accesses are acted on. This indicates an important implicit relationship between Δ, *abound*, and *sbound*. We chose our constants so that the schedule never froze.

The most surprising result of our experiments is that when annealing was used with TR and $\mathrm{MPF}_{k(t)}$ no improvement was found using Δ_1 or Δ_2. In fact when Δ grew rapidly the performance could be much worse. This is counterintuitive. It seems that if the same number of accesses are acted on then increasing the number of failures should not hurt, especially when the latter accesses were thought to be poor candidates for moves. (In the limit using Δ_2 should give better results; however it is very slow since we used up to 44000 accesses for 100 records!)

The resolution of this paradox is twofold. In the early stages denying any moves slows convergence. The second reason is that TR and $\mathrm{MPF}_{k(t)}$ are very good to start with. In particular in the later stages candidates for moves will only move a short distance so it is not appropriate to "protect" against moves from the rear of the list. Such moves tend to fine-tune the arrangement.

In contrast, annealing is helpful for MTF. The primary reason MTF performs so poorly is that in the later stages it does not protect itself against ill-advised moves. For the linear data, with $\delta = .8, b_1 = .01, b_2 = 5/n$, we found a reduction in the percentage of optimal from 121.70 to 106.65 using Δ_2. Similarly for the Zipf data, with $\delta = .8, b_1 = 3.0, b_2 = 5/n$, we found a reduction from 140.70 to 107.99. However Δ_1 did not do as well. For example, for the latter case with $\delta = .8, a_1 = .6, a_2 = 0$, we found a reduction from 140.70 to only 112.68. (The constants were chosen above to give the best observed performances and to involve the same number of accesses.)

We noted that the amount of processing for $\delta = .9$ was double that for $\delta = .8$ with little improvement. This, of course, is a function of the small n; as n increases δ must as well to get comparable results.

5. Annealing f-Lists

We now turn to the original problem of reorganizing the records of an f-list. Due to the expense of these experiments, which used a larger n, we relied on the nearly one thousand trials we did for linear lists to dictate what we tried for f-lists. In general our results were quite favorable, supporting the hypothesis that effective rules for the two problems should be similar.

We did not use random data since real data for an f-list was available (a disk in this case) [CARS85]. We did our experiments with a file that recorded 94834 consecutive seeks on a disk with $n = 842$ cylinders. We used the file to generate accesses, rewinding as often as necessary. With $\delta = .8$ and .9 we used, at least, 176820 and 370480 accesses respectively. The average access time was computed before and after reorganization. The initial average was, perhaps, overly large due to the placement of highly used directory records in the low numbered cylinders, as is common. Our measure of performance was the percent of improvement.

The same program was used with the following small modification. The position *pos* of an access, used in the formulas of section 4, was the absolute distance from the previous access and *moverule* always moved a record towards the previous record as if it were the "front".

We began again with $\Delta = 0$ and varied k. For $\delta = .8$ and $k=1$ (MTF), 100, and 842 (TR) the improvement was 49.7%, 32.6%, and 47.2% respectively. The surprising result is that TR was not best after so many accesses. We feel this is due to the large number of infrequently accessed records and the large distances they must travel. With $k=100$ records move faster but it is still somewhat "TR-like" for nearby records where stability matters. Note that these findings are not in agreement with the list case as seen in figure 2, where $k=12$ is approximately in the same ratio to n as $k=100$ is here. This probably indicates a crossover has yet to be reached.

When the time varying $k = k(t)$ was tried, with $\delta = .8$, we had a 34.7% reduction. While this was much better than TR it was still a bit worse than with $k=100$. From our experience above we could predict further improvement by having our rule become "TR-like" more slowly. This motivated a *moverule* that moves a record $\lfloor pos/k(t) \rfloor$ positions, where

$$k(t) = \left\lfloor \sigma(t)^{\epsilon} n \right\rfloor, \qquad \epsilon \geq 1.$$

As ϵ increases $k(t)$ becomes more slow growing. We found for the above trial with $\epsilon=3$ we had a 31.5% improvement, narrowly beating $k=100$.

Judging from the lack of success with $\Delta > 0$ for linear lists we were skeptical of their utility for f-lists. Again, in all cases for Δ_1 and Δ_2 we had a degradation of performance. The degradation increased when, by adjusting the constants, the algorithm became more selective. This supports the analysis in the previous section that argued annealing would not improve on $\mathrm{MPF}_{k(t)}$ in the short run. No attempt was made to gauge the improvement over unannealed MTF for f-lists.

Finally we tried $\delta = .9$. For $k = 100$ the improvement was 26.3%. (Note that for the larger n in this section this increase in δ is significant.) For $k = k(t)$ and $\epsilon = 3$ we found a 28.2% improvement, but with $\epsilon = 10$ we had 25.9%. This indicates that ϵ should be time varying as well; this was not explored.

6. Conclusions

The primary finding is that for self-organizing lists and f-lists when annealing is restricted to use $O(1)$ extra storage the benefits are not as favorable as when good self-organization is used alone. A preliminary comparison of our results with some annealing results, using TR only, for the same disk data, using $O(n)$ and $O(n^2)$ extra storage [CARS85] indicates that our rules are superior in time and performance. This indicates that improving self-organization techniques is more profitable than annealing. More work needs to be done to explain why pure self-organization rules are so robust that, in the short run, annealing seems to interfere with their performance. Further work is planned that will provide a full statistical comparison of our algorithms and those in [CARS85] as well as some others in the literature. Extending the model to allow several independent (or dependent) fingers has not been explored. Other algorithms for the f-list problem have been recommended that simply use local improvement [CARS85, FLOR78, RAMA71].

An open problem is to apply these techniques to data structures that support key searchs, more sophisticated than linear search. The case of binary search trees [ALLE78, BITN79, GONN81] is somewhat anomalous since *move-to-root* is quite good but the analogue of TR is very bad. Can time-varying rules and annealing mediate this anomaly? Using a more complicated *splaying* rule a "snapshot" theorem for static arrangements has been derived that promises good performance [SLEA85b].

7. Acknowledgements

I would like to acknowledge Scott Carson and Paul Reynolds Jr. for introducing this problem to me and the use of their disk data.

8. References

[ALLE78] B. Allen and I. Munro, Self-organizing binary search trees, *J. ACM*, **25**, 1978, pp. 526-535.

[ANDE82] A. E. Anderson, P. Nash and R. R. Weber, A counterexample to a conjecture on optimal list ordering, *J. Appl. Prob.*, **19**, 1982, pp. 730-732.

[BENT85] J. L. Bentley and C. C. McGeoch, Amortized analyses of self-organizing sequential search techniques, *Comm. ACM*, **28**, 1985, pp. 404-411.

[BITN79] J. R. Bitner, Heuristics that dynamically organize data structures, *SIAM J. Computing*, **8**, 1979, pp. 82-110.

[BURV73] P. J. Burville and J. F. C. Kingman, On a model for storage and search, *J. Appl. Prob.*, **10**, 1973, pp. 697-701.

[CARS85] S. Carson and P. Reynolds Jr., Adaptive reorganization of disk cylinder structures, Computer Science Tech. Rep. 85-20, Dept. Computer Science, Univ. Virginia , 1985.

[CHUN85] F. R. K. Chung, D. J. Hajela and P. D. Seymour, Self-organizing sequential search and Hilbert's inequalities, *Proc. 17th ACM Symp. Theory Comp.*, 1985, pp. 217-223.

[FLOR78] A. Flory, J. Gunther and J. Kouloumdjian, Data base reorganization by clustering methods, *Information Systems*, **3**, 1978, pp. 59-62.

[GARE79] M. R. Garey and D. S. Johnson, *Computers and Intractability*, Freeman, 1979.

[GONN81] G. Gonnet, J. I. Munro and H. Suwanda, Exegesis of self-organizing linear search, *SIAM J. Computing*, **10**, 1981, pp. 613-637.

[HEND76] W. J. Hendricks, An account of self-organizing systems, *SIAM J. Computing*, **5**, 1976, pp. 715-723.

[KAN80] Y. C. Kan and S. M. Ross, Optimal list order under partial memory constraints, *J. Appl. Prob.*, **17**, 1980, pp. 1004-1015.

[KIRK83] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, *Science*, **220**, 1983, pp. 671-680.

[KONN81] L. K. Konneker and Y. L. Varol, A note on heuristics for dynamic organization of data structures, *Info. Proc. Let.*, **12**, 1981, pp. 213-216.

[LAM81] K. Lam, M. K. Siu and C. T. Yu, A generalized counter scheme, *Theoretical Comp. Sci.*, **16**, 1981, pp. 271-278.

[LAM84] K. Lam, M. Leung and M. Siu, Self-organizing files with dependent accesses, *J. Applied Prob.*, **21**, 1984, pp. 343-359.

[MCCA65] J. McCabe, On serial files with relocatable records, *Operations Research*, **13**, 1965, pp. 609-618.

[OOMM84] B. J. Oommen, On the use of smoothsort and stochastic move-to-front operations for optimal list reorganization, *Proc. 23rd Allerton Conf.*, 1984, pp. 243-251.

[RAMA71] C. V. Ramamoorthy and P. R. Blevins, Arranging frequency dependent data on sequential memories, *AFIPS Conf. Proc.: Spring Joint Comp. Conf.*, **38**, 1971, pp. 545-556.

[RIVE76] R. Rivest, On self-organizing sequential search heuristics, *Comm. ACM*, **19**, 1976, pp. 63-67.

[SLEA85a] D. D. Sleator and R. E. Tarjan, Amortized efficiency of list update and paging rules, *Comm. ACM*, **28**, 1985, pp. 202-208.

[SLEA85b] D. D. Sleator and R. E. Tarjan, Self-adjusting binary search trees, *J. ACM*, **32**, 1985, pp. 652-686.

[TENE78] A. Tenenbaum, Simulations of dynamic sequential search algorithms, *Comm. ACM*, **21**, 1978, pp. 790-791.

[TENE82] A. M. Tenenbaum and R. M. Nemes, Two spectra of self-organizing sequential search algorithms, *SIAM J. Computing*, **11**, 1982, pp. 557-566.

[VAQU80] A. Vaquero and J. M. Troya, Placement of records on linear storage devices, in *Information Processing 80 (IFIP)*, S. Lavington (ed.), 1980, 331-336.

[WONG83] C. K. Wong, *Algorithmic Studies in Mass Storage Systems*, Computer Science Press, 1983.

[YU79] C. T. Yu, M. K. Siu, K. Lam and M. Ozsoyoglu, Performance analysis of three related assignment problems, in *Proc. ACM-Sigmod Intl. Conf. on Management of Data*, P. Bernstein (ed.), 1979, 82-91.

AN IMPLEMENTATION OF ORDERED PRIORITY QUEUES

SHOU-HSUAN STEPHEN HUANG
Department of Computer Science
University of Houston-University Park
Houston, Texas 77004

ABSTRACT

A new data structure called ordered priority queue is introduced in this paper. Elements stored in the data structure have a primary order (key) and a secondary order (priority) associated with them. An ordered min-priority queue allows users to find the minimum priority element in any range (according to key order) in $O(\log n)$ time. Such a data structure with n elements can be created in $O(n \log n)$ time using $O(n)$ storage. A specific implementation based on median split trees is presented. Sequential access of the elements can be done in $O(n \log \log n)$ time and $O(\log n)$ extra storage.

1. INTRODUCTION

Given a set of elements, each with a key value and a priority, a priority queue ([1], [2]) is a data structure which permits the following operations to be performed on such a set:

1. Find the element with the minimum priority,
2. Delete the minimum priority element, and
3. Insert a new element.

In addition to the efficient execution of the above operations, it is also important that we be able to build a priority queue efficiently given n elements. Several representations of this data structure are known, including an efficient representation via a heap ([7]).

The following definition of heap is taken from ([5]). A (min-)heap is a completely balanced binary tree of height h in which all leaves are at a distance h or h-1 from the root and all descendants of a node have a greater priority than that of the node; furthermore, all leaves at level h are as far left as possible.

In most cases, the key values of the elements form a "natural" order. This natural order of the elements are ignored in priority queues. Thus we cannot easily retrieve information such as the minimum priority element in a given range according the key values. We shall consider a new data structure which allows the elements to be "ordered" in addition to having priorities.

Elements stored in the data structure have a primary order, called keys, and a secondary order, called priorities. Keys and priorities are selected from two (possibly different) linearly ordered sets. Without loss of generosity, we shall use positive integers for both keys and priorities in this paper. A typical query to the structure looks like: "find the element between key1 and key2 with the minimum priority" where key1 and key2 are two parameters. Such a query will be called a range priority query from now on.

The above query can certainly be answered in O(n) time by scanning the whole list of elements. However if we are going to perform such queries many times, say n times, then it may pay to do some preprocessing to cut down the cost. The cost to perform n range priority queries, in this case, is O(n**2).

The second solution is to construct a 2-dimensional table of the minimum priority elements for all possible pairs of keys. With such a table, range priority query can be answered in constant time. However the cost for this approach is at least O(n**2) in preprocessing time and O(n**2) in storage.

In some applications, we may want to delete the minimum priority element depending on some conditions. The second approach suggested above cannot be used in this situation unless we update the table whenever deletion occurs.

A new data structure is defined below. Given a set of elements, an (minimum) ordered priority queue is a priority queue which also supports range priority queries. Our implementation of ordered priority queue has the following features:

1. the time to find the minimum priority element is O(1),
2. the time to delete the minimum priority element is O(log n),
3. the time to find the minimum priority element in a range (lower, upper) is O(log n),
4. the time to delete the minimum priority element in a range (lower, upper) is O(log n),
5. the time to access all elements sequentially is O(n log log n),
6. the structure can be created in O(n log n) time,
7. the storage requirement is O(n).

Thus it takes no more than O(n log n) time (including the cost of initialization) to perform n range priority queries. We can allow new elements to be inserted into the data structure, however the insertions may increase the cost of the above operations. So ordered priority queues are considered static data structures even though deletions are allowed.

Throughout the paper n is an integer of the form 2**k - 1 and all log´s are base 2. This restriction on the number of elements can be removed easily. A symmetric maximum ordered priority queue can be constructed in a symmetric way.

Section 2 presents an implementation of priority queues via ordered heaps. Algorithms to access and update ordered heaps are given in Section 3. An implementation without using pointers are given in Section 4 together with conclusions.

2. AN IMPLEMENTATION

It is easy to see that a simple binary search tree, which requires the key of the root to be between the keys of the two subtrees, is not able to implement a priority queue. In order to implement ordered priority queues discribed in Section 1, we have to generalize the definition of heaps.

Definition 1: A completely balanced binary tree T of n elements (each with key value and a priority) is called an ordered (min-) heap (OH) if it satisfies the following conditions:

1. The root has a priority less than or equal to those of all elements in S.
2. All elements, except the root, are divided into two subtrees L and R which are ordered heaps themselves.
3. All elements in the left subtree L have smaller key values than those elements in R. The largest key value in L is called the partition value or split value of the root of the tree.

A completely balanced binary tree is a binary tree with minimum external path length ([5]). Such a tree has the property that all leaves appears on at most two consecutive levels. Notice that in the definition, we did not require the leaves on the bottom level to be as left as possible (as in the case of an ordinary heap).

An ordered heap can be viewed as a variation of median split tree. Split trees ([6]) are binary trees with the following properties. Each node of the tree has an additional "split value" stored in the node so that all elements less than or equal to the split value are stored in the left subtree and all elements that are greater than the split value are stored in the right subtree. The subtrees are themselves split trees. In addition, the element with the highest access frequency are stored in the root of the tree. A formal definition of the split trees can be found in ([3]). Median split tree chooses the split values such that the two subtrees are balanced in terms of the number of elements. It can be shown that such a tree can be constructed in O(n log n) time ([6]).

The priority in an ordered heap is equivalent to the access frequence of split trees. We select the root element to be the one with the minimum priority. The split value of the root is selected (initially) to be the median of the rest of the elements, i. e., as balanced as possible. An example (with n = 15) is given in Figure 1(A) below. Given a list of n elements, an ordered heap can be built in O(n log n) time with O(n) storage.

3. ORDERED HEAP ALGORITHMS

Since the element with minimum priority is in the root of the tree, it can be found in constant time. Since the height of such a balanced tree is $\lceil \log (n+1) \rceil$, we can delete the minimum priority element (of the tree) and update the tree structure in O(log n) time. However, the reorganization of the ordered heap is slightly different from that of an ordinary heap. The detail can be found in a more general case below. Searching for the minimum priority element in a range can be done in O(log n) time since there are at most two paths (from the root to the leaves) which must be checked. An outline of the algorithm is given below.

```
function find-min (root: ptr; low, hi: element): ptr;
{assume key values are  of element type}
var  minleft, minright: ptr;
begin
   if root = nil
      then find-min := nil          {CASE 1: empty subtree}
   else if (low <= root^.key) and (root^.key <= hi)
      then find-min := root                        {CASE 2}
   else if root^.split >= hi                       {CASE 3}
      then find-min := find-min (root^.left, low, hi)
   else if root^.split < low                       {CASE 4}
      then find-min := find-min (root^.right, low, hi)
   else begin                                      {CASE 5}
      minleft := find-min (root^.left, low, hi);
      minright := find-min (root^.right, low, hi);
      if minleft^.priority < minright^.priority
         then find-min := minleft
         else find-min := minright;
   end;
end;
```

Since there are two recursions in the above algorithm, it is possible that the algorithm have to search most of the binary tree in order to find the minimum priority element in a given range. However we can show that this is not the case. When we execute CASE 5 for the second time (or later), even though there are two recursions to continue, one of them (the "inside" recursion) will terminate by checking only one more node (i. e., terminate at CASE 2). The "outside" recursion may propagate further down the tree.

In the example given in Figure 1(A), to find the minimum priority element in the range of Key 8 to Key 12, we need only to visit 8 nodes (nodes H, D, F, G, L, J, N, and M). Subtree rooted at node J is an "inside" subtree (i. e., there are search pathes on both sides of the subtree), so the minimum priority element of the subtree is in the root node J. Subtrees rooted at L and H are "outside" subtrees, the recursion may continue down the tree.

Lemma 1: Algorithm Find-Min visits at most $4\lceil \log (n+1)\rceil - 5$ nodes in an ordered heap of n elements.

Proof: From the above observation, we know that the algorithm visits at most 4 nodes per level. Thus the cost is no more than $4\lceil \log (n+1)\rceil -5$. []

The following algorithm deletes an element from an ordered heap. The procedure is called using the pointer returned by function Find-Min.

```
-----------------------------------------------------------------
procedure delete (var p: ptr);
{p points to a node to be deleted from the ordered heap}
begin
   if      (p^.left = nil)  then p := p^.right
   else if (p^.right = nil) then p := p^.left
   else if (p^.left^.pri < p^.right^.pri) then begin
      p^.element := P^.left^.element;
      delete (p^.left);
   end else begin
      p^.element := p^.right^.element;
      delete (p^.right);
   end;
end;
-----------------------------------------------------------------
```

Another interesting feature of ordered heaps is that they can be accessed sequentially in O(n log log n) time with only O(log n) extra storage. An algorithm, using an ordinary priority queue of size $\lceil \log (n+1) \rceil$ is given below.

```
-----------------------------------------------------------------
Procedure Sequential (root: ptr);
var pq: pqueue;       {a regular priority queue}
     i: integer;
     e: element;
begin
   initialize (pq);
   i := 0;
   visit (root, i);
   while not empty (pq) do begin
      deletemin (pq, e);
      output (e);
   end;
end;

Procedure visit (root: ptr, var i: integer);
{Postorder traversal}
begin
   if root <> nil then begin
      i := i+1;                  {inserting the i-th time}
      if i <= h {height of the tree} then {no deletion}
      else begin
         deletemin (pq, e);
         output (e);
      end;
      e^.pri := root^.key;
      insert (e, pq);
      visit (root^.left, i);
      visit (root^.right, i);
   end;
end;
-----------------------------------------------------------------
```

In the algorithm, we first put h = $\lceil \log (n+1) \rceil$ elements along the leftmost path of the tree into the priority queue. Notice that we are using the key values of T as the priority for the (regular) priority queue pq. Procedure Visit is, basically, a recursive preorder traversal algorithm. It deletes the minimum priority element from pq and inserts the next (in terms of the preorder ordering) element into the priority queue. Finally, the algorithm outputs all the elements left in the priority queue.

Theorem 1: The above algorithm outputs elements in ascending (key) order from an ordered heap in O(n log log n) time using O(log n) extra storage.

Proof:
(Correctness of Algorithm Sequential) We have to show that when the algorithm is about to output an element, say e, from the priority queue pq, all elements less than e must have been output before. We divide the nodes in the tree (i. e., ordered heap) T into five disjoint sets: (see Figure 2)

A = {all ancestors of e},
SL = {all decendent of e in the left subtree},
SR = {all decendent of e in the right subtree},
L = {all nodes to the left of A and SL}, and
R = {all nodes to the right of A and SR}.

Let e´ be any element not output but with a smaller key value than that of e. We may also assume that e´ is not in the priority queue pq at this time. Clearly e´ cannot be in R due to the definition of the ordered heap. If e´ is in L, then it must be in the priority queue pq. Thus we should output e´ instead of e according to the definition of priority queue. For the same reason, e´ cannot be in A. Thus e´ can only be in SL or SR if such e´ exists.

Note that all elements in L have smaller key values than that of e. Thus none of them are in the priority queue pq at that particular time. Yet the priority queue pq contains exactly h elements except at the beginning and the end of the algorithm. So some of the h elements must come from sets SL, SR or R. Since there are less than h elements in the path from the root to e´ (not including e´), there is at least one element e" in pq which is to the left (i. e. has a smaller key value) of e´. Thus key(e") < key(e´) < key(e). This contradict to the fact that e is the smallest element in pq. So there is no such e´ since it cannot be in any of the five sets.

(Complexity of the Algorithm) We have to insert (and delete) each element into pq once. Since the size of the priority queue is $h = \lceil \log (n+1) \rceil$, the cost for each insertion and deletion is O(log log n). []

Since log log n is fairly small for practical purpose, we claim that the elements are "almost" sorted.

4. AN IMPLICIT IMPLEMENTATION AND CONCLUSION

Finally, we can store the whole tree (ordered heap) in an array A without using pointers. There are two ways of doing it. One way is follow the implicit implementation of an ordinary heap by storing the left (right) son of a node A[i] in A[2i] (A[2i+1], respectively). The second way is to put the root of the tree in A[n/2]. All elements less than the split value of the root is stored in the lower half of the array recursively. The other half of the elements are stored in the upper half of the array recursively. Thus the only extra storage is in the space for the n split values. The second array implementation of the ordered heap in Figure 1(A) is shown in Figure 1(B). All the algorithms discussed in the previous section can be modified to avoid the use of pointers.

We have presented a new data structure that has several advantages over the traditional priority queues. An ordered heap can be viewed as a combination of a binary search tree and an ordinary priority queue. A table comparing these two structure is given below. There are several ways to use this new data structure. For example, we can use it to store points in a two dimensional space and ask questions like: "what is the point with the minimum x-coordinate between y1 and y2?" In fact, the study of this data structure is motivated by the study of optimal binary split trees ([3], [4]). In the construction of the optimal trees, we have to find (n times) the elements with the maximum access frequency in a given range.

One thing this structure does not do well is the insertion of new elements as we discussed in Section 1. We can certainly insert a new element and reorganize the heap so that the definition is not violated. However, the best algorithm that we have so far requires O(n) time thus not very practical.

Table 1: A Comparison of Priority Queues (PQ) and
Ordered Priority Queues (OPQ)

Operations	PQ	OPQ
Find-Min	O(1)	O(1)
Delete-Min	O(log n)	O(log n)
Find-Min (Range)	O(n)	O(log n)
Delete-Min (Range)	O(n)	O(log n)
Creation	O(n)	O(n log n)
Sequential Access	O(n**2)	O(n log log n)
Storage Cost	O(n)	O(n)

References

[1] M. D. Atkinson, J.-R. Sack, N. Santoro and Th. Strothotte: An efficient, implicit double-ended priority queue, Carleton University Technical Report SCS-TR-55, 1984.

[2] D. E. Knuth: The art of computer programming, Vol. 3: Sorting and Searching, Addison-Wesley, 1973.

[3] S.-H. S. Huang and C. K. Wong: Optimal binary split trees, J. Algorithms, 5, 69-79, 1984.

[4] S.-H. S. Huang and C. K. Wong: A faster algorithm to construct optimal binary split trees, Technical Report UH-CS-84-12, University of Houston, 1984.

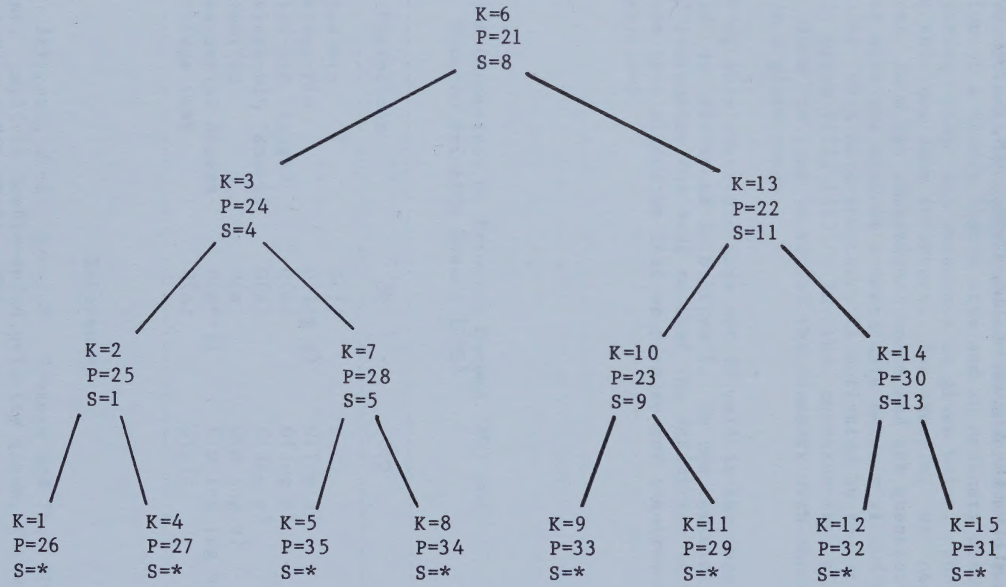
[5] E. M. Reingold, J. Nievergelt and N. Deo: Combinatorial Algorithms, Prentice-Hall, 1977.

[6] B. A. Sheil: Median split trees: A fast lookup technique for frequently occuring keys, Comm. ACM 21, 947-958, 1978.

[7] J. W. J. Williams: Algorithm 232, Comm. ACM, 7, 347-348, 1964.

Figure 1: An Ordered Priority Queue with 15 elements.
(K = Key, P = Priority, and S = Split value)

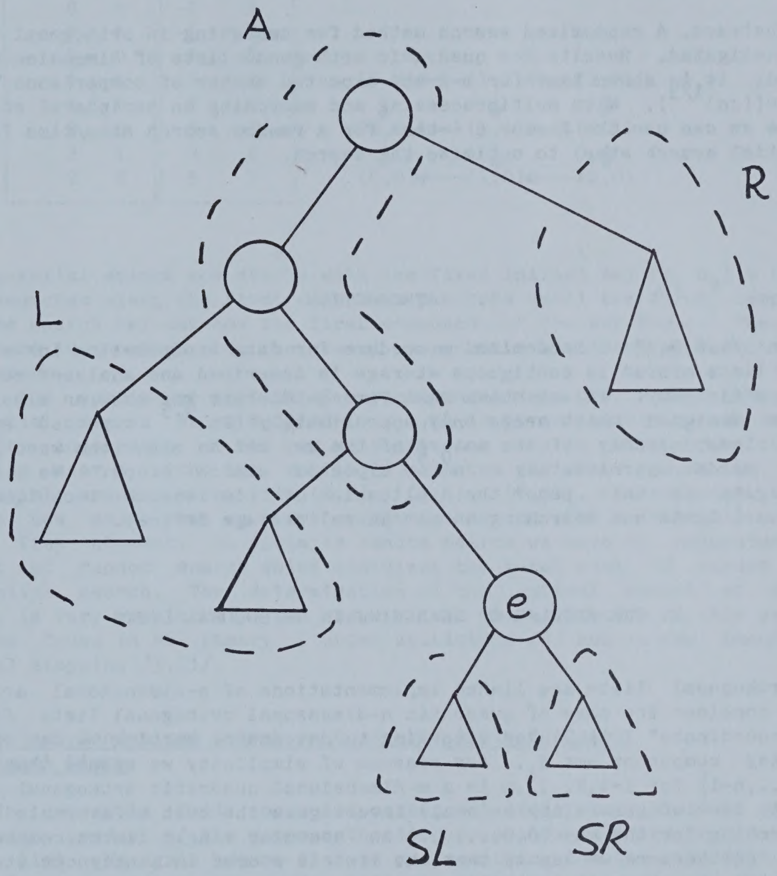
(A)


K=6
P=21
S=8
K=3
P=24
S=4
K=13
P=22
S=11
K=2
P=25
S=1
K=7
P=28
S=5
K=10
P=23
S=9
K=14
P=30
S=13
K=1
P=26
S=*
K=4
P=27
S=*
K=5
P=35
S=*
K=8
P=34
S=*
K=9
P=33
S=*
K=11
P=29
S=*
K=12
P=32
S=*
K=15
P=31
S=*

(B)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Key	1	2	4	3	5	7	8	6	9	10	11	13	12	14	15
Priority	26	25	27	24	35	28	34	21	33	23	29	22	30	32	31
Split	*	1	*	4	*	5	*	8	*	9	*	11	*	13	*

Figure 2: The classifications of nodes in an ordered heap.



RANDOMIZED SEARCH IN ORTHOGONAL LISTS AND ON PERIPHERAL STORAGE DEVICES

WOLFGANG H. JANKO
Inst. für Angew. Betriebswirtschaftslehre/Universität Karlsruhe
D-7500 Karlsruhe 1 / FRG

Abstract. A randomized search method for searching in orthogonal lists is investigated. Results for quadratic orthogonal lists of dimension m are derived; it is shown that for m=2 the expected number of comparisons is of order $O((cn)^{1/3})$. With multiprocessing and searching on peripheral storage devices we can use the factor c (=time for a random search step/time for a sequential search step) to optimize the search.

1. INTRODUCTION

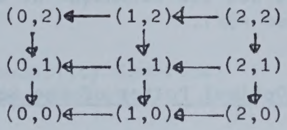
In /1,8,9,10/ a randomized procedure for data manipulation in sorted linked lists stored in contiguous storage is described and analysed relativ to its efficiency. It was shown that for equidistant key sets an algorithm can be designed which needs only approximately $(2n)^{0.5}$ expected search steps. Independently of the nature of the key set an algorithm was given which needs approximately $2n^{0.5}$ expected search steps. We shall investigate in this paper the application of this search technique to orthogonal lists and searching on peripheral storage devices.

2. THE PROBLEM OF SEARCHING IN ORTHOGONAL LISTS

Orthogonal lists are linked implementations of n-dimensional arrays. Let us consider the case of quadratic n-dimensonal orthogonal lists first. Each "coordinate" i is linked according to descending sorting order of the i-th key component set S_i. For reasons of simplicity we assume that $S_i = \{0,1,\dots,n-1\}$ for i=1,2,...,m in a m-dimensional quadratic orthogonal list. Without loss of generality we shall investigate the most unfavorable case of searching for the key (0,0,....,0) and assuming singly linked components first. Furthermore we assume that the list is stored in contiguous storage (analogous to the linear list version). The data structure used to represent a n x n x.....x n m-dimensional orthogonal list is a n^m x 2m array; for example let $S_i = \{0,1,2,\}$ for i=1,2 and use the locations 0,1,...,.35 for storage using nine blocks of 4 cells each; (the pointer value of the last component in descending sorting order is -1):

block location i	key components		pointers	
0	0	0	-1	-1
1	0	1	-1	0
3	1	0	0	-1
4	1	1	1	3
5	1	2	2	4
6	2	0	3	-1
7	2	1	4	6
8	2	2	5	7

Grid representation of the orthogonal list:



A sequential search now starts with the fixed initial key $(s_1\ s_2) = (2\ \ 2)$ and searches along the first component pointers until the first component of the search key matches the first component of the key found. Then this procedure is repeated with the second component. Similarly we search in m dimensions.

Using the special properties of our component sets the total number of sequential search steps to find (0,...,0) is given exactly by the sum of the key components of the key we start our search with. For example starting search with the key (1,2) means we can find (0,0) in three sequential search steps (via (0,2), (0,1), (0,0)). Introducing random search now means that we start sequential search with a randomly chosen first list element. To optimize random search we have to determine the amount of random search which minimizes the total cost of random plus sequential search. The determination of the optimal amount of random search is very sensitive to the restrictions given. Answers to this problem can be found in the theory of order statistics /4/ and in the theory of optimal stopping /3,11/.

2.1 Optimal Randomized Search in Quadratic Fully Occupied Orthogonal Lists

In quadratic fully occupied orthogonal lists every element of the cartesian product of the component sets corresponds to a key in the orthogonal list. We again assume that we are searching for the key (0,0,...0) only. In random search we assume uniformly distributed access. As we assume that the component sets are all the same and equal to $S_1 = \{0,1,2,..., n-1\}$ the distance in sequential search steps is equal to the sum of the key components. The problem of determining random search optimally is equivalent to the problem of drawing randomly from the set of key component sums with replacement at the cost of c to minimize the value drawn plus sampling cost with $c=t_r/t_s$; t_r denotes the (average) time for a random search step and t_s denotes the (average) time for a sequential search step. (c is such a multiplier to convert random search step cost to sequential search step cost using time units.)

2.1.1 Models to Optimize Random Search

There are at least two different classes of models to optimize random search:

a) sequential models ("sequential" is usually used in the statistical sense in chapter 2.1.1)

b) nonsequential models

Sequential models are the most efficient ones. We still have to distinguish between models using a maximum number of observations in random search and models not using such a limit. The latter ones are more efficient and easy to implement as there is only one stopping value v^* which has to be used. In the more restricted policy with a given maximum number of random search steps k we have to use the technique of backward induction to construct the optimal policy (see /3/).

2.1.1.1 The Optimal Policy of the Sequential Model

Let us consider the transform:

$$T(v) = \sum_{j=0}^{v} (v-j)p(j) \quad \text{for } v \in N_0 \text{ and } j=0,1,\ldots,m(n-1)$$

where p(j) denotes the probability of drawing a component sum of j at random search. This transform T(v) is for p(j)>0 (for every j) a strictly increasing piecewise linear function in $[0,m(n-1)]$. Let us now denote by v^* the expected value of sequential search steps if we continue random search optimally. We stop random search than for a value of $j \leq v^*$ and continue for $j > v^*$. v^* now is determined by

$$v = E(\min\{j,v\})+c = \sum_{j=0}^{\lfloor v \rfloor} j p(j) + \sum_{j=\lfloor v \rfloor +1}^{m(n-1)} v\, p(j) + c$$

$$= \sum_{j=0}^{\lfloor v \rfloor} j\, p(j) + \sum_{j=0}^{m(n-1)} v\, p(j) - \sum_{j=0}^{\lfloor v \rfloor} v\, p(j) + c = v + \sum_{j=0}^{\lfloor v \rfloor} (j-v)p(j) + c$$

Therefore we get $c = \sum_{j=0}^{v} (v-j)p(j) = T(v)$

which determines the optimal policy v^* completely. If we introduce a maximum number of random search steps we have to distinguish between policies with and without recall. These policies consist of a vector of stopping intervals /3/. As these policies are less efficient as the one given above and induce considerable additional cost they shall not be considered here. The same is true for the class of rank oriented models, which use as only information source the relative rank of the number of sequential search steps found at the last random search step within the sorted numbers of sequential search steps found in random search before /7/. The only alternative model which is of interest is the nonsequential model of search, where the optimal amount of random search is entirely determined in advance by techniques used in the theory of order statistics.

2.1.1.2 The Optimal Amount of Random Search with the Non-Sequential Sampling Policy

This policy is determined by the minimum of the sum consisting of the expected minimum of a sample out of a population with the known distribution (of "sequential" search steps) plus the cost of random search (expressed in units of "sequential" search steps). We such have to determine u(k), that is the expected value of the sample minimum as function of the sample size ($a_{1,k}$) plus the sample cost function kc:

$u(k)=E(\min\{x_1,x_2,\dots,x_k\})+kc=a_{1,k}+kc$

As $a_{1,k}$ is monotonically decreasing and kc is linearly increasing with k we end up determining this value of k where u(k) is minimal. For an arbitrary discrete distribution with the probability function p(j) the value $a_{1,k}$ is determined by (see /4/, pp. 35):

$$a_{1,k} = \sum_{i=0}^{m(n-1)} (1-P(X\leq i))^k$$

(P(X≤i) denotes the cumulative probability function)

The optimal value of k is such determined by

$$a_{1,k-1}-a_{1,k} > c \text{ and } a_{1,k}-a_{1,k+1} \leq c.$$

As $a_{1,k}$ is monotonically decreasing with decreasing decrements, k^* is unique.(For m=1 and the dicrete uniform distribution on $\{0,1,\dots,n-1\}$ we can immediately see that

$$a_{1,k}=n^{-k} \sum_{i=0}^{n-1} i^k$$

For m=1 explicit results for sampling without replacement are given in Janko /7/).

2.1.2. The Distribution of Sequential Search Steps Searching Randomly in Singly Linked Orthogonal Lists

The number of component sums having a value of j for j=0,1,...,m(n-1) is most easily determined using the related composition or cell occupancy problem. Our problem is such equivalent to the problem to add up m numbers out of $\{1,2,\dots,n\}$ such that their sum is equal to j+m. This is equal to the problem of distributing j balls in m cells such that each cell contains not more than n-1 balls. The generating function (see /5/) for this problem is

$$(t^1+t^2+\dots+t^n)^m = t^m(1+t+\dots+t^{n-1})^m = t^m\left(\frac{1-t^n}{1-t}\right)^m = t^m(1-t^n)^m(1-t)^{-m}$$

Such we get for

$$p_m(j)= \sum_{k=0}^{\lfloor j/n\rfloor} (-1)^k \binom{m}{k}\binom{j+m-nk-1}{j-nk} /n^m$$

for the distribution of sequential search steps.

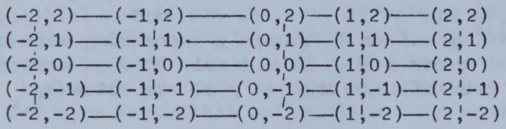
For m=2 this formula degenerates to

$$p_2(j)=\begin{cases}(1+j)/n^2 & \text{for } 0\leq j\leq n-1\\ (2n-j-1)/n^2 & \text{for } n-1< j\leq 2(n-1)\end{cases}$$

2.1.3. The Distribution of Sequential Search Steps With Doubly Linked Orthogonal Lists

In doubly linked orthogonal lists we assume at first that the cardinality of the component set is an odd natural number. Let us furthermore assume that the component set is equal to $\{-(n-1)/2, -(n-3)/2,\dots\dots, -1,0,1,\dots,(n-1)/2\}$.

Grid model for n=5 and m=2:



Searching now for the element with the key (0,0,...,0) using always the shortest way means that adding up the absolute value of the key components gives exactly the number of sequential search steps to find the search key. The problem is equivalent to the problem of distributing j (j=0,1,...,m(n-1)/2) balls on m cells where each cell in addition can bear one of the signs (+,-). Considering that each cell can at most hold only (n-1)/2 balls we need at least $\lceil 2j/(n-1) \rceil$ cells but have at most m cells to distribute j balls.

Now let us denote by q_u $(j/s_i \geq 1;\ i=1,2,....,u)$ the number of possibilities to distribute j balls such that u cells have at least one ball. Having m cells in total there are (m-u) cells without balls or in total there are $\binom{m}{u}$ such possibilities to draw u cells out of m possible cells.

At most we have m cells holding at least one ball. From the u cells having at least one ball, each cell can have one of two signs. In total there are 2^u possibilities to distribute the signs. We get therefore for the number of elements having a distance of exactly j sequential search steps in the m-dimensional list with doubly linked components:

$$q_m(j) = \sum_{i=r}^{\min\{m,j\}} 2^i q_i(j/s_{k_l} \geq 1;\ l=1,2,...,i)\binom{m}{i}$$

with $r = \lceil 2j/(n-1) \rceil$.

As we know from single linkage that (s_{k_l} denotes the number of balls in the k_l-th cell)

$$q_i\ (j/s_{k_l} \geq 1;\ l=1,2,....,i) = \sum_{k=0}^{\lfloor 2(j-i)/(n-1) \rfloor} (-1)^k \binom{i}{k} \binom{j-k(n-1)/2-1}{j-k(n-1)/2-i}$$

For the probability p(j) of choosing randomly a distance of j in a random search step in a doubly linked m-dimensional orthogonal list (m is odd) we get therefore:

$$p_m(j) = n^{-m}\, q_m(j)$$

This formula degenerates for m=2 to

$$p_m(j) = \begin{cases} 4(n-j) & \text{for } (n-1)/2 < j \leq n-1 \\ 4j & \text{for } 0 < j \leq (n-1)/2 \\ 1 & \text{for } j=0 \end{cases}$$

2.1.4 The Distribution of Sequential Search Steps at Even Cardinality of the Component Set in Doubly Linked Orthogonal Lists

If the value of n is even we can use recursion formulas to calculate the number of list elements having a sequential search distance of j steps. If the cardinality of the component set is even we consider search for

(0,0,...,0) with the component set $\{-(n-2)/2, -(n-4)/2, \ldots, 0, 1, 2, \ldots, (n-2)/2, n/2\}$

Lemma: The recursive formula for m=1 is

$$q_1(j) = \begin{cases} 1 & j=0 \\ 2 & j>0 \text{ and } j<n/2 \\ 1 & j=n/2 \end{cases}$$

and for arbitrary m and even n we get:

$$q_m(j) = \begin{cases} q_{m-1}(j) + q_{m-1}(j-n/2) + 2 \sum_{i=j-(n-2)/2}^{j-1} q_{m-1}(i) & \text{for } n/2 \leq j \leq mn/2 \\ q_{m-1}(j) + 2 \sum_{i=0}^{j-1} q_{m-1}(i) & \text{for } 0<j \leq n/2-1 \\ 1 & \text{for } j=0 \end{cases}$$

To proof this recursion formula we have to consider the fact that the additional m-th component can either be 0, n/2 or 1,2,...,(n-2)/2 with the + or - sign in the latter case. Taking into consideration that a component value of n/2 is not possible if $j \leq n/2-1$ we have to treat this case separately. The probability of having exactly j search steps starting with a randomly drawn element is equal to

$p_m(j) = q_m(j)/n^m$

2.1.5 The Optimal Sequential Policies for m=2

Using the discrete distribution $p_m(j)$ of sequential search steps in random search we determine the expected value of the optimal stopping policy and the stopping value v together. The expected value of the non-sequential policy can be calculated easily but a closed form expression was not derived.

For the sequential policy we get for c=T(v) and singly linked lists (for a different derivation see /6/):

$$v \sum_{i=0}^{v} p(i) - \sum_{i=0}^{v} ip(i) = c \text{ for } p(i) = (1+i)/n^2$$

From $\sum_{i=0}^{v} p(i) = (v+1)(v+2)/(2n^2)$ for $v \leq n-1$ we get

$$v(v+1)(v+2)/(2n^2) - v(v+1)(v+2)/(3n^2) = c$$

or $v(v+1)(v+2) = 6n^2c$

We such have to solve $v^3+3v^2+2v-6n^2c=0$ for v.

Using Cardano`s formula /2/ we get for $v \leq n-1$ for $s = (9n^4c^2-1/27)^{1/2}$ and for $9n^4c^2 > 1/27$ the one real root:

$$v = (3n^2c+s)^{1/3} + (3n^2c-s)^{1/3} - 1$$

which is very close to $(6n^2c)^{1/3} - 1$.

Let us consider now doubly linked orthogonal lists. From

$$v \sum_{i=0}^{v} p(i) - \sum_{i=0}^{v} ip(i) = c$$ we get for $p(i) = 4i/n^2$ $(i>0)$ and

$p(i) = n^{-2}$ for i=0 with $v \leq \lfloor (n-1)/2 \rfloor$ the expression

$$4vn^{-2} \sum_{i=1}^{v} i + v/n^2 - 4n^{-2} \sum_{i=1}^{v} i^2 = c \text{ getting } v^3+v/2-1.5n^2=0$$

Using again Cardano`s formula we get for $s=(d^2/4+1/216)^{1/2}$ and $d=1.5n^2c$ the one real root:

$v= (d/2+s)^{1/3} + (d/2-s)^{1/3}$ which is very close to $(1.5n^2c)^{1/3}$.

2.2 Random Search for Arbitrary Keys in Non-Quadratic Orthogonal Lists and with Different Key Component-Sets

Using the generating function for possibly different cardinalities n_j for $j=1,2,..,m$ of the component sets we get for singly linked lists:

$$\prod_{i=1}^{m} \left(\sum_{j=0}^{n_i-1} t^j \right)$$

for the calculation of the exact distribution of the number of keys having exactly a distance of j from the key searched for. The case of doubly linked orthogonal lists can than be treated similarly or by Bayesian methods (see /11/) using the sequential policy. Throughout we assumed that the component sets are equal to $\{0,1,...,n_j-1\}$ for some value of n_j and $j=1,2,...,m$. If this is not the case we need at least component sets where the numerical difference between any two neighbours in sorting order is constant for every component set to choose exactly that randomly drawn element which is closest to the search key or closer than v_* sequential search steps. That this is in principle possible with great accuracy for almost every realistic key set is shown in /8/ using the transformation of the key component values with their empirical distribution function or an approximation thereof. Searching for arbitrary keys the results given here can always be guaranteed by using a distance function (see /6,8/) and circular linkage independent of the actual probabilities of the data instances. As distance function we use $d^*(s_g,s_j) = \sum_{k=1}^{m} d(s_{g_k},s_{j_k})$.

3. RANDOM SEARCH ON PERIPHERAL STORAGE DEVICES AND USING MORE THAN ONE PROCESSOR

While searching on discs and other similar peripheral storage devices holding lists in random physical order on contiguous storage we have to take into consideration the different access time for random and sequential search. Whereas sequential search is very time consuming, for random search the list elements can usually be read in physical order and blockwise. Thus the factor c can be considerably (10^{-2} to 10^{-4}). Similarly if we have k processors the average time for a random search step per key can approximately be k^{-1} times the sequential search time as we can compare one random key with k search keys simultaneously.

4. CONCLUSIONS

Searching in linear lists stored in contiguous storage can be considered as a special case (m=1) of searching in m-dimensional orthogonal lists. For quadratic m-dimensional orthogonal lists the distribution $p_m(j)$ for j=0, 1,.., (n-1)m sequential search steps at singly linked lists and for j=0,1,.., m(n-1)/2 search steps at doubly linked lists could be derived for choosing the starting point randomly. The nonsequential policy which is very useful for m=1 seems not to be useful for $m > 1$, because we have to transform the key component set always to an - at least approximately in its expectation /8/ - equidistant set to be able to recognize the key closest to the key searched for in random search. But in this case application of the less costly sequential policy of random search is applicable. If we optimize the sample size prior to search or optimize the stopping interval used taking into consideration the different average costs of random and sequential search steps this search method allows us to tune randomized search for the application on peripheral storage devices and for using more than one processor. Results not discussed here in detail are the experience with this search method having not fully occupied orthogonal lists and using Bayesian stopping methods (learning randomized search) and variations thereof. The experimental results indicate that randomized search can be used successfully with sparsely occupied lists and special provisions have to be taken in the case of frequent deletions and insertions. For batched search this method seems to be very useful too. More detailed results will be presented in another paper.

Acknowledgement: I am indebted to Mrs. Moritz for typing the paper and to Mr. Timpe for pointing my attention to Cardano`s work.

REFERENCES

/1/ Bentley, J.L., Stanat, D.F., Steele, J.M.: Analysis of a Randomized Data Structure for Representing Ordered Sets, 19th Annual Allerton Conf. on Comm., Control and Comp., Allerton House, Monticello, (Ill.), 364-373, (1981).

/2/ Cardano, G., Artis Magnae sive de Regulis Algebraicis, Lib. unus (1545) in: The Great Art or the Rules of Algebra, Witmer T.R., (ed. and translated), The MIT Press, Cambridge, Mass., 1968.

/3/ Chow, Y.S., Robbins, H., Siegmund D.: Great Expertations: The Theory of Optimal Stopping, Houghton Mifflin Comp., Boston, 1971.

/4/ David, H.A.: Order Statistics, John Wiley & Sons, New York, 1970.

/5/ Eisen, M.: Elementary Combinatorial Analysis, Gordon and Breach, New York, 1969.

/6/ Janko, W.: Optimale Stichprobensuchverfahren bei Implementierung von Datenbeständen mittels orthogonaler Listen.
Teil I: Die Implementierung von Datenbeständen mittels orthogonaler Listen.
Teil II: Optimale Stichprobensuche in orthogonalen Listen, Angew. Informatik 11,12/73, (1973).

/7/ Janko, W.: Searching for the Lowest Offer When the Distribution of Offers is Truly Unknown, in: Contributions to Econometrics and Statistics Today, H. Schneeweiß, H. Strecker (Eds.), Springer-Verlag, Heidelberg, 1985.

/8/ Janko, W.: An Insertion Sort for Uniformly Distributed Keys Based on

Stopping Theory, Proc. Int. Comput. Symp., Morlet E., Ribbens D. (Eds.), North Holland, 373-379 (1977).

/9/ Leighton,T., Lepley, M.: Probabilistic Searching in Sorted Linked Lists, 20th Allerton Conf. on Comm., Control and Comput., Monticello, Ill., 500-506 (1982).

/10/ Nevalainen, O.: On a Statistical Access Method for a Non-Dense List Area, Angew. Inf., 1/75, 9-12, (1976).

/11/ Randolph, P.H.: Optimal Stopping Rules for Multinomial Observations, Metrika, Vol. 16, 48-61, (1968).

FREE DISTANCES OF CONTINUOUS PHASE MODULATIONS UNDER EXTREME FILTERING

N. SESHADRI AND J. B. ANDERSON
ECSE Department
Rensselaer Polytechnic Institute
Troy, New York 12180-3590

ABSTRACT

We evaluate the minimum distance and hence the error probability of continuous phase modulations that have been subjected to extreme channel filtering. Using a frequency domain interpretation of the signal space distance, we show that under extreme bandlimitations, the error event that is most likely to cause the minimum distance is characterized by a spectral null at d.c. Among other results, we show that either reducing the modulation index or increasing the partial response length does not result in improved performance.

I. INTRODUCTION

In practice, modulations have to be filtered before transmission because of their infinitely wide spectrum. It is well known that filtering results in intersymbol interference. The effect of the intersymbol interference on continuous phase modulations (CPM) was first studied by Aulin and Sundberg [1], who modeled the channel filter as one with a short impulse response. On the other hand, we assume that the filter is strictly bandlimited. The filtered CPM signal is then optimally detected after being corrupted by additive white Gaussian noise. We confine our studies to the evaluation of the normalized minimum Euclidean distance, which is a measure of the large signal to noise ratio error performance of any modulation scheme.

Our earlier works [2,3] regarding the effect of channel filtering on the error performance of CPM assumed that the one-sided bandwidth - symbol duration product (BT) was greater than or equal to 0.25. In this work, we study the error performance of filtered CPM at narrow filter bandwidths, i.e., BT < 0.25. As an example, we concentrate on BT = 0.1. After a search for the error events causing the minimum distance at these bandwidths, a theory is developed to characterize these error events, which leads to a reduced search for the minimum distance causing error event.

II. FILTERED CPM

Let $\underline{a}$ be an infinitely long data sequence that has to be transmitted, with a data symbol a_i taking on any one of M possible values ± 1, ± 3, ..., $\pm (M-1)$ at random. In this work, we concentrate on binary signalling, i.e., $M = 2$. The CPM signal, obtained as a result of modulation of the phase by the data sequence $\underline{a}$, can be written as

$s(t,\underline{a}) = \sqrt{2E/T}\cos(2\pi f_c t + \phi(t,\underline{a}) + \phi_0)$, where f_c is the carrier phase, and $\phi(t,a) = 2\pi h \sum a_i q(t-iT)$. E is the symbol energy, and T is the symbol duration. ϕ_0 is the initial phase, which is set to zero under the assumption of coherent detection. We define the phase q(t) as

$$q(t) = \int_0^T g(x)dx \quad , \quad t \geq 0 \quad ,$$

where g(t) is a positive, symmetric and causal frequency pulse, lasting L symbol durations. For L equal to 1, we have full-response signalling, and for L greater than 1, we have partial response signalling. In general, L is a positive integer. The frequency pulse g(t) is chosen in such a way that $q(t) = 1/2 \quad , \quad t \geq LT$.

When a CPM signal is filtered by a channel with impulse response h(t) $= h_c(t) \cos 2\pi f_c t - h_s(t) \sin 2\pi f_c t$, the resulting filtered CPM signal is given by

$$\begin{aligned} s_0(t,\underline{a}) &= s(t,\underline{a}) * h(t) \\ &= i_0(t,\underline{a}) \cos 2\pi f_c t - q_0(t,\underline{a}) \sin 2\pi f_c t \end{aligned}$$

where

$$i_0(t,\underline{a}) = \cos \phi(t,\underline{a}) * h_c(t) - \sin \phi(t,\underline{a}) * h_s(t)$$

and

$$q_0(t,\underline{a}) = \cos \phi(t,\underline{a}) * h_s(t) + \sin \phi(t,\underline{a}) * h_c(t)$$

where * denotes convolution.

The probability that a transmitted signal $s_0(t,\underline{a})$ is detected as another signal $s_0(t,\underline{b})$, where $\underline{a}$ and $\underline{b}$ differ at least in one symbol position is given by

$$p_e = Q\left(\sqrt{D^2/2N_0} \right) \quad ,$$

where

$$Q(x) = 1/\sqrt{2\pi} \int_x^{\infty} e^{-y^2/2} \, dy$$

and

$$D^2 = \int_{-\infty}^{+\infty} | s_0(t,\underline{a}) - s_0(t,\underline{b}) |^2 \, dt \quad .$$

The quantity $N_0/2$ is the power spectral density of the additive white Gaussian noise. At high signal to noise ratio, the probability that a symbol is erroneously detected is dominated by the minimum squared signal space distance which is defined as

$$D^2_{min} = \inf_{\substack{\underline{a}, \underline{b} \\ a_k \neq b_k \\ a_i = b_i , \; i < k}} \int_{-\infty}^{+\infty} | s_0(t,\underline{a}) - s_0(t,\underline{b}) |^2 dt$$

We define the normalized minimum distance as

$$d^2_{min} = D^2_{min}/2E_b \quad ,$$

where E_b is the average energy per bit transmitted after filtering. This normalized distance enables us to compare the performance of different CPM schemes that transmit at different energy levels.

III. DISTANCE CALCULATIONS FOR FILTERED CPM

Provided $f_c T \gg 1$, we can write the squared signal space distance between $s_0(t,\underline{a})$ and $s_0(t,\underline{b})$ as

$$D^2 = \sqrt{E/T} \int_{-\infty}^{\infty} |i_0(t,\underline{a}) - i_0(t,\underline{b})|^2 \; dt$$

$$+ \sqrt{E/T} \int_{-\infty}^{\infty} |q_0(t,\underline{a}) - q_0(t,\underline{b})|^2 \; dt \quad . \tag{1}$$

When the channel impulse response is finite, it is possible to evaluate the minimum distance using the double dynamic programming search as developed by Anderson and Taylor [4] among others, or use the tree pruning algorithm developed by Aulin et. al. [5]. However, when the channel is strictly bandlimited, the distance accumulated during any interval is dependent on the entire data sequences $\underline{a}$ and $\underline{b}$. Hence, we cannot use any of the methods mentioned for calculating the minimum distance.

In [2] and [3], we have used an alternative strategy to evaluate the distance. This is based on the Parseval's relation, which can be used to express (1) as

$$D^2 = \int_{-\infty}^{\infty} |H(f)|^2 \; |S_a(f) - S_b(f)|^2 \; df \quad , \tag{2}$$

where $H(f) = F\{h(t)\}$,

$S_a(f) = F\{s(t,\underline{a})\}$,

$S_b(f) = F\{s(t,\underline{b})\}$,

with F denoting the Fourier transform.

$S_a(f) - S_b(f)$ can be obtained for any two distinct CPM signals $s(t,\underline{a}) - s(t,\underline{b})$ differing over a short duration simply by Fourier transforming their difference. Such a pair of CPM signals constitute an unfiltered error event, which in future will be referred to simply as an error event. The filtered Euclidean distance can be obtained with great accuracy since H(f) has a narrow effective support. Due to the infinite length of the intersymbol interference in the situation we are considering, one in theory has to perform an exhaustive search to evaluate the minimum distance. Short of that, an upper bound is obtained. However, by careful consideration of the search, we can obtain the minimum distance. Based on (2), we can interpret the squared signal space distance as the energy associated with the given error event within the pass-band of the filter.

The results of the minimum distance obtained, using the procedure described, are summarized in figures 1 and 2 for 1-RC and 3-RC modulation, where L-RC means that the frequence pulse g(t) is a raised cosine pulse, lasting L symbol durations. In general, we have observed that at moderate bandlimitations, $BT \geq 0.25$, the error event causing the minimum distance is typically the one that causes the minimum distance for no bandlimitation. For binary CPM, except for certain larger values of the modulation index (see [5] and [6]), this error event is characterized by

$$a_i = b_i \quad , \quad i \neq 0,\, 1 \quad ,$$

$$a_0 = -b_0 = +1, \text{ and } a_i = -b_1 = -1 \quad . \tag{3}$$

At severe bandlimitation, we find that the error event causing the minimum distance is different from the one described in (3). For example, for 1-RC modulation, the minimum distance causing error event is characterized by

$$\underline{a} = \;..\,,\; a_{-1} + 1 \;,\; - 1 \;,\; -1,\; + 1 \;,\; a_4 \;....$$

$$\underline{b} = \ldots, b_{-1}\ -1, +1, +1, -1, b_4 \ldots,$$

$$a_i = b_i \quad , \quad i \neq 0, 1, 2, 3 \quad . \qquad (4)$$

On observing this result carefully, we find that $\underline{b}$ is obtained by reading $\underline{a}$ and reversing the sign, i.e., $a_0 = -b_3$, $a_1 = -b_2$ etc.

This interesting result is intuitively justified by considering the spectrum of the error event constituted by $s(t,\underline{a})$ and $s(t,\underline{b})$ with $\underline{a}$ and $\underline{b}$ given by (4). This error event has a d.c. spectral null, i.e.,

$$| S_a(0) - S_b(0) | = 0 \quad .$$

Since this event exhibits a d.c. spectral null, this event is likely to have less difference energy compared to an error event without a d.c. null, when the filter band is very narrow. The spectra of the error events causing the minimum distance at severe bandlimitions and moderate bandlimitation are shown for 1-RC in figures 4 and 3. In the following sections, this intuitive justification is made mathematically rigorous.

IV. ERROR EVENTS WITH D.C. NULL

Expressed in terms of the baseband low-pass signals, a CPM signal modulated by a data sequence $\underline{a}$ can be written as

$$s(t,\underline{a}) = \sqrt{2E/T}\ [\cos \phi(t,\underline{a}) + j \sin \phi(t,\underline{a})] \quad ,$$

where $j = \sqrt{-1}$. Similarly, the CPM signal modulated by the data sequence $\underline{b}$ can be written as

$$s(t,\underline{b}) = \sqrt{2E/T}\ [\cos \phi(t,\underline{b}) + j \sin \phi(t,\underline{b})] \quad .$$

Let

$$I_a(f) = F\ \{\cos \phi(t,\underline{a})\}\ ,\ Q_a(f) = F\ \{\sin \phi(t,\underline{a})\}$$

$$I_b(f) = F\ \{\cos \phi(t,\underline{b})\}\ ,\ Q_b(f) = F\ \{\sin \phi(t,\underline{b})\} \qquad (5)$$

The condition for the d.c. spectral null in terms of the Fourier transforms in (5) can be written as

$$| [I_a(0) - I_b(0)] + j\ [Q_a(0) - Q_b(0)] | = 0$$

which implies that

$$I_a(0) - I_b(0) = 0 \qquad (6)$$

$$Q_a(0) - Q_b(0) = 0 \quad . \qquad (7)$$

In the time domain, (6) and (7) will imply that

$$\int_{-\infty}^{+\infty} [\cos \phi(t,\underline{a}) - \cos \phi(t,\underline{b})]\ dt = 0$$

and

$$\int_{-\infty}^{+\infty} [\sin \phi(t,\underline{a}) - \sin \phi(t,\underline{b})]\ dt = 0$$

Further simplification results in

$$\int_{-\infty}^{+\infty} \sin\frac{[\phi(t,\underline{a})+\phi(t,\underline{b})]}{2}\ \sin\frac{[\phi(t,\underline{a})-\phi(t,\underline{b})]}{2}\,dt = 0, \tag{8}$$

$$\int_{-\infty}^{+\infty} \cos\frac{[\phi(t,\underline{a})+\phi(t,\underline{b})]}{2}\ \sin\frac{[\phi(t,\underline{a})-\phi(t,b)]}{2}\,dt = 0 \tag{9}$$

Let $\phi(t,\underline{a}) + \phi(t,\underline{b})$ be the sum phase trajectory, and $\phi(t,\underline{a}) - \phi(t,\underline{b})$ be the difference phase trajectory. We will now state conditions for the d.c. null in terms of the difference and sum phase trajectories. In general, it is difficult to find all the solutions to (8) and (9), but we can derive a sufficient condition. It can be seen from (8) and (9) that for a d.c. null:

a) The sum phase trajectory should be symmetric about a time $t = xT$, if the error event lasts from $t = 0$ to $t = 2xT$
b) The difference phase trajectory should be anti-symmetric about the same point $t = xT$.

These two conditions will mean that for the L.H.S. of (8),

$$\int_{0}^{xT} \sin\frac{[\phi(t,\underline{a})+\phi(t,\underline{b})]}{2}\ \sin\frac{[\phi(t,\underline{a})-\phi(t,\underline{b})]}{2}\,dt$$

$$= -\int_{xT}^{2xT} \sin\frac{[\phi(t,\underline{a})+\phi(t,\underline{b})]}{2}\ \sin\frac{[\phi(t,\underline{a})-\phi(t,\underline{b})]}{2}\,dt .$$

A similar relation holds true for the L.H.S. of (9).

To gain further insight into the problem, we now express these conditions in terms of the data sequences $\underline{a}$ and $\underline{b}$, which modulate the phase of the CPM signals $s(t,\underline{a})$ and $s(t,\underline{b})$, respectively.

Full Response Signalling

Without loss of generality in the calculation of the minimum distance, let us assume that $\underline{a}$ is the transmitted sequence and $\underline{b}$ is the error sequence, with $\underline{a}$ differing from $\underline{b}$ for the first time at $t = 0$. This implies that all data prior to $t = 0$ were correctly decoded. Then, it can be shown that the accumulated phase $\emptyset(t,\underline{a})$ and $\emptyset(t,\underline{b})$ at $t = 0$ does not figure in the minimum distance calculations, and what actually matters in the minimum distance calculations are the last L−1 data symbols prior to $t = 0$. We thus set the accumulated phase to zero at $t = 0$ without loss of generality.

With the assumption that the accumulated phases are zero at $t = 0$, we can now state the conditions for the anti-symmetry of the difference phase without actually proving them.

Theorem Let $\gamma_i = a_i - b_i$, and N be a positive integer. Then, for the difference phase trajectory to be anti-symmetric,

a) $\sum_{i=0}^{N-1} \gamma_i = 0$

b) $\gamma_i = \gamma_{2N-i-1}$, $0 \leq i \leq 2N-1$

c) $\gamma_i = 0$, $i \notin [0, 2N-1]$. (10)

We first note that for full-response signalling, error events causing a d.c. null last an even number of symbol durations, 2NT. Condition (a) and (b) together imply the anti-symmetry of the difference phase with a zero cross over at time t = NT. The zero cross over can occur only at the end of a symbol interval due to the nature of the frequency pulse chosen. Condition (c) ensures that the error event starts at t = 0 and ends at t = 2NT.

Similarly, for the sum phase trajectory to be symmetric, if $a_i + b_i = \eta_i$, then we have

$$\eta_i = \eta_{2N-i-1} \quad , \quad 0 \le i \le 2N-1 \; . \tag{11}$$

To find sequences $\underline{a}$ and $\underline{b}$, satisfying (10) and (11), we assume that the difference phase trajectory is anti-symmetric. When γ_i is non-zero, there is no ambiguity in determining a_i and b_i. For example when i = +2(-2), a_i = + 1(-1) and b_i = -1(+1). The problem is to choose a_i and b_i when γ_i is zero. Here we have two possible choices, both a_i and b_i are either +1 or -1. However, (11) restricts the choice of a_i and b_i. For (11) to hold, we must choose

$$a_i = -a_{2N-i-1} \; .$$

Thus if $a_i = +1$, then necessarily,

$$a_{2N-i-1} = -1.$$

Thus when γ_i is zero in K positions, we have totally $2^{K/2}$ choices of $\underline{a}$ and $\underline{b}$. For example, if

$$\underline{\gamma} = +2,\ 0,\ -2,\ -2,\ 0,\ +2$$

then $\underline{a}$ is either

$$+1,\ -1,\ -1,\ -1,\ +1,\ +1 \tag{12}$$

or

$$+1,\ +1,\ -1,\ -1, -1,\ +1 \; . \tag{13}$$

Corresponding to (12), we have

$$\underline{b} = -1,\ -1,\ +1,\ +1,\ +1,\ -1 \; ,$$

for the error event to have a d.c. null.

The interesting observation here is that $\underline{b}$ is obtained by reading $\underline{a}$ backwards with the sign of every a_i reversed. If

$$\phi(t,\underline{a}) = 2\pi h \sum a_i \, q(t-iT) \; ,$$

then during the interval [0, 2NT],

$$\phi(t,\underline{b}) = 2\pi h \sum -a_{2N-1-i} \, q(t-iT),$$

which after simplification reduces to

$$\phi(t,\underline{b}) = \phi(2NT-t,\underline{a}) \quad t \varepsilon [0,2NT] \; .$$

Thus, the phase of the CPM signal $s(t,\underline{b})$ is obtained by reflecting the phase trajectory of $s(t,\underline{a})$ about the phase axis, and then delaying it by 2NT seconds. The relation among the inphase and quadrature components of the CPM signal turns out to be

$$i(t,\underline{b}) = i(2NT-t,\ \underline{a}) \qquad t\varepsilon[0,2NT] \ .$$

$$q(t,\underline{b}) = i(2NT-t,\ \underline{b}) \qquad t\varepsilon[0,2NT] \ .$$

With this result, one can prove that a d.c. null occurs by simply taking the Fourier transforms of the inphase and quadrature components and evaluating them at f = 0. For the transforms, we have

$$I_b(f) = I_a(f)\ e^{-j2\pi fNT} \ ,$$

$$Q_b(f) = Q_a(f)\ e^{-j2\pi fNT} \ ;$$

at f = 0 ,

$$I_a(0) = I_b(0) \ ,\ Q_a(0) = Q_b(0) \ .$$

This results in a spectral null by (6) and (7).

Partial-Response Signalling

For partial response signalling, similar results hold as for full response, i.e.,

$$b_i = -a_{N-i-1} \ , \ -L+1 \leq i \leq N-1$$

where N is a positive integer. However, unlike full response signalling, the data pre-history has to be taken into account. For example, consider 3-RC modulation with the error event consisting of sequences

$$\begin{array}{lllllllllll}
 & & & a_{-2} & a_{-1} & a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\
\underline{a} & = & \ldots\ldots, & +1, & -1, & +1, & -1, & -1, & +1, & +1, & -1, \ \ldots \\
\underline{b} & = & \ldots\ldots, & +1, & -1, & -1, & +1, & +1, & -1, & +1, & -1, \ \ldots \ .
\end{array} \qquad (14)$$

a_{-2} and a_{-1} constitute the data pre-history. This is the error event that causes the minimum distance at BT = 0.1. We note that $b_i = a_{5-i}$, $-2 \leq i \leq 5$, and hence this error event has a d.c. null. However, if we consider the error event with a different prehistory, viz,

$$\begin{array}{lllllllllll}
\underline{a} & = & \ldots\ldots, & +1, & +1, & +1, & -1, & -1, & +1, & +1, & -1, \ \ldots \\
\underline{b} & = & \ldots\ldots, & +1, & +1, & -1, & +1, & +1, & -1, & +1, & -1 \ \ \ldots \quad ,
\end{array} \qquad (15)$$

we find that this error event has only a near d.c. null, and that its distance is greater than the distance of the error event given by (14) at BT = 0.1.

V. TAYLOR SERIES ANALYSIS OF ERROR EVENTS WITH D.C. NULLS

In this section a Taylor series analysis of a specific error event is made and a relationship is obtained between the distance, bandwidth and error event length. While the analysis is done for just one event, useful conclusions can be drawn based on the results obtained.

Consider the series expansion of an error event spectrum $S(f)$ about $f = 0$. Let

$$S(f) = \sqrt{2E/T} \quad [I(f) + jQ(f)]$$

where $I(f) = F\{\cos \phi(t,\underline{a}) - \cos \phi(t,\underline{b})\}$,

$$Q(f) = F\{\sin \phi(t,\underline{a}) - \sin \phi(t,\underline{b})\} .$$

When $S(0) = 0$, both $I(0) = 0$ and $Q(0) = 0$.

Further, the first derivative of $I(f)$ and $Q(f)$ must be zero at $f = 0$ by the symmetry of the null. Using this fact, we can approximate $I(f)$ and $Q(f)$ as

$$I(f) \simeq I''(0) \quad (2\pi f)^2/2! \qquad Q(f) \simeq Q''(0) \quad (2\pi f)^2/2!$$

where $I''(0)$ is the second derivative of $I(f)$ at $f = 0$ and similarly for $Q''(0)$. These are given by the second moment expressions

$$I''(0) = -4\pi^2 \int_{-\infty}^{+\infty} t^2 \, [\cos \phi(t,\underline{a}) - \cos \phi(t,\underline{b})] \, dt \tag{16}$$

$$Q''(0) = -4\pi^2 \int_{-\infty}^{+\infty} t^2 \, [\sin \phi(t,\underline{a}) - \sin \phi(t,\underline{b})] \, dt \tag{17}$$

Based on (16) and (17), it can be said that in general long error events tend to have large values of $I''(0)$ and $Q''(0)$. To gain further insight, we consider continuous phase FSK (or '1-REC' signalling [5]). At narrow bandwidth, the error event causing the minimum distance is given by

$$\underline{a} = \ldots., a_{-1}, +1, -1, -1, +1, a_4, \ldots$$

$$\underline{b} = \ldots., b_{-1}, -1, +1, +1, -1, b_4, \ldots$$

$$a_i = b_i \quad , \quad 0 > i > 3 \quad .$$

For this error event, it is seen that $I''(0) = 0$.

Hence

$$|S(f)| = \sqrt{2E/T} \quad |Q''(0)| \quad (2\pi f)^2/2! \quad ,$$

where

$$Q''(0) = -8\pi^2 \int_0^{4T} t^2 \sin \phi(t,\underline{a}) \, dt \quad .$$

For small values of h, $\sin \phi(t,\underline{a}) \simeq \phi(t,\underline{a})$.

Using this approximation, we get

$$Q''(0) = 64\pi^3 h T^3 \quad .$$

The squared signal space distance is

$$D^2 = E \, [32\pi^3 h]^2 \, (BT)^5 \quad .$$

For $h = 0.125$ and $BT = 0.1$, we get $D^2 = 0.0307E$, compared to the actual value of $D^2 = 0.0281E$.

We observe that the quantity $Q''(0)$ and $I''(0)$ is generally dominated by the last few intervals of an error event, and behaves as $O(N^3)$, where NT is the length of the error event. In particular, in the example we are considering, the squared signal space distance solely due to the last of the 4 intervals is approximately 0.015E, which is half the squared distance due to the entire error event.

VI. CONCLUSIONS

A frequency domain technique for evaluating tight estimates of the minimum distance was presented at extreme filter bandwidths, the error event causing the minimum distance has a d.c. spectral null. It was also shown that a large class of error events having this property is characterized by the fact that the error data sequence $\underline{b}$ is obtained by reading the transmitted sequence $\underline{a}$ backwards and reversing the signs. This fact has enabled us to drastically reduce the search for the error events causing the minimum distance.

Further, using a Taylor series approximation, we have analyzed specific error events which lead us to conclude that among the error events with d.c. spectral null, the very short duration error events have less distance compared to longer error events. This reduces the event search even more. We have also shown that partial-response smoothing tends to perform worse than full response signalling and that reducing the modulation index leads to worse performance as well.

REFERENCES

[1] T. Aulin, and C-E Sundberg, "Minimum Distances for Optimum and Suboptimum Receivers for Bandpass Filtered CPM," Technical Report TR-153, Telecommunications Theory Department, University of Lund, Lund, Sweden, August 1981.

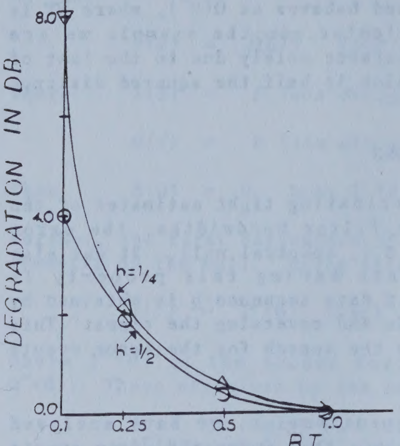
[2] N. Seshadri, and J. B. Anderson, "Heavily Filtered Continuous Phase Modulations," Proceedings, 1985 Conference On Information Sciences and Systems, The Johns Hopkins University, Baltimore, Maryland, March 1985.

[3] N. Seshadri, and J. B. Anderson, "Minimum Distances Of Heavily Filtered Continuous Phase Modulations," 1985 IEEE International Symposium On Information Theory, Brighton, England, June 1985.

[4] J. B. Anderson and D. P. Taylor, "A bandwidth-efficient class of signal space codes," IEEE Trans. Information Theory, IT-24, 703-712, Nov. 1978.

[5] T. Aulin, N. Rydbeck and C.-E. Sundberg, "Continuous Phase Modulation - Parts I and II: Partial Response Signaling," IEEE Trans. Commun., COM-29, 196-225, March 1981.

This work was supported in part by the
Motorola Foundation, Schaumburg, IL.



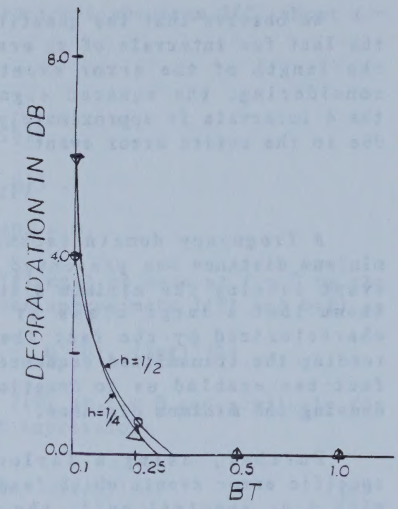
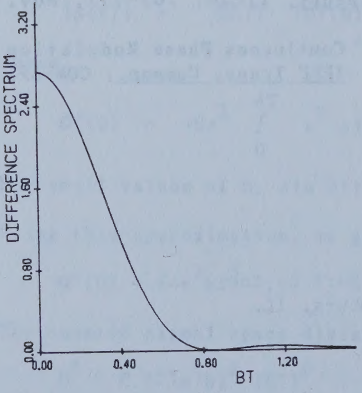


FIG. 1: Performance degradation due to filtering for 1-RC CPM.

Fig. 2: Performance degradation for 3-RC CPM.



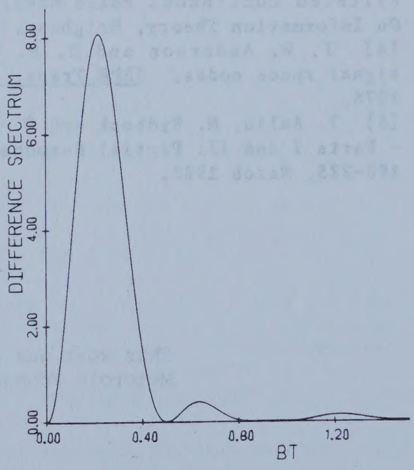


Fig. 3: Spectrum of the error event leading to d_{min} for moderate bandlimitation of 1-RC.

Fig. 4: Spectrum of the error event leading to d_{min} for severe bandlimitation of 1-RC.

PERFORMANCE OF DIVERSITY/REED-SOLOMON CODED SSFH COMMUNICATIONS WITH ERRORS/ERASURES VIA RATIO THRESHOLDING

K.G. CASTOR
Electrical Engineering Department
Air Force Inst. of Tech.
Wright-Patterson AFB, OH 45433

W.E. STARK
EECS Department
Univ. of Michigan
Ann Arbor, MI 48109

ABSTRACT

Ratio thresholding is used to develop symbol erasure criteria for an M-ary orthogonal FSK spread-spectrum frequency-hopping system. Performance is analyzed for systems using L-diversity transmission, Reed-Solomon coding, and parallel decoding with incomplete and complete inner decoders. Two inner decoding schemes are described and results presented for partial-band Gaussian noise interference with a (32,16) Reed-Solomon code at a bit error rate of 10^{-4}; $L = 3,7$; and various erasure threshold values.

I. INTRODUCTION

In this paper we present an approach for the design of spread-spectrum frequency-hopped (SSFH) systems in partial-band interference without side information. The system we propose develops a qualitative measure of the jamming environment by using a ratio threshold test like that introduced by Viterbi [1]. Concatenated diversity/Reed-Solomon coding is used in a parallel decoding method method similar to that proposed by Stark and Pursley[2]. Criteria for symbol erasure in the inner decoder are developed using the results of threshold testing. Other researchers have developed decoding algorithms for the channel without side information, however the performance measures have usually been that of channel cutoff rate[1, 3]. The channel cutoff rate is useful in determining the bit error rate when convolutional coding is employed. However, for diversity and Reed-Solomon coding the bit error rate must be calculated directly.

For our analysis we consider a system using an M-ary orthogonal signal set, frequency-hopping and noncoherent reception in partial-band Gaussian interference. Each symbol is received with every E_s in a noise environment which consists of jamming of power $\rho N_J / 2$ in a fraction ρ of the spread-spectrum bandwidth. The remainder of the bandwidth is modeled as noiseless. The performance measure of interest will be decoded bit error probability.

In Section II, we describe the general parallel decoding algorithm with specific descriptions of the thresholding and decoding methods. The system analyses are outlined in Section III and in Section IV the system performance is compared with that for a similar parallel decoding system which assumes known jammer state.

II. DECODING ALGORITHMS

The parallel decoding system is shown in Figure 1. The inputs, E_i, $i = 0, M-1$ are the outputs of M energy detectors matched respectively to the orthogonal M-ary signal set; system synchronization and dehopping are assumed. The M-ary symbol detector and threshold test unit produces a pair of outputs $(\hat{s}, q)$ for each M-ary input. The first element, $\hat{s} \in \{0,1,\ldots,M-1\}$ is a hard decision estimate of the transmitted symbol corresponding to $E_{\max} = \max\{E_i\}$. The second element, q, is a binary soft decision metric resulting from a threshold test on the E_i values.[1]. If θ, the threshold, is chosen so that $\theta \geq 1$ then q is given by

$$q_i = \begin{cases} 0 & \text{if } E_{\max} \geq \theta E_i \quad i = 0, \ldots, M-1 \; ; \; i \neq \max \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

With L-diversity transmission, the input to each of the two inner decoders is an L-tuple of pairs $\{(\hat{s}_j, q_j); \; j = 1, \ldots, L\}$. The incomplete decoder outputs either a symbol estimate or an erasure for each L-tuple of pairs. On the other hand, the complete decoder makes a hard decision estimate (no erasures) of the transmitted symbol. Assuming an (N, K) Reed-Solomon code, N outputs of the incomplete decoder are input to the Reed-Solomon decoder to produce K information symbols. With bounded distance decoding, the Reed-Solomon decoder will, with high probability, fail rather than produce an erroneous output. Whenever the decoder does not fail, the system output is the output of the Reed-Solomon decoder. In the event of decoder failure, the system output is the information symbol portion of the complete decoder output.

In this paper we consider two algorithms for the inner decoders. The first algorithm, referred to as algorithm A, is as follows. The A incomplete decoder outputs an erasure if either or both of the following two conditions occur:

I: No symbol of the set $\{\hat{s}_j; \; j = 1, \ldots, L\}$ occurs more then $\lfloor L/2 \rfloor$ times. (L assumed odd)

II: The sum of the corresponding L quality bits satisfies:

$$\sum_{j=1}^{L} q_j > \theta_q \tag{2}$$

where $\theta_q \in (\lceil L/2 \rceil, \ldots, L\}$ is a second threshold value.

Otherwise, the A incomplete decoder outputs that symbol which occurred the majority of the time. The A complete decoder ignores the quality bits and makes a maximum likelihood decision based on the L symbol estimates $\{\hat{s}_j; \; j = 1, \ldots, L\}$.

The incomplete decoder for the second algorithm, B, is a two step decoder. First, each symbol estimate having $q_i = 1$ is replaced with an erasure. Second, the decoder outputs an erasure if no non-erased symbol occurs more then $\lfloor L/2 \rfloor$ times. As a consequence if an erasure occurs the majority of the time then the incomplete decoder will also erase. If some (nonerased) symbol occurs more than $\lfloor L/2 \rfloor$ times then the B incomplete decoder outputs the non-erased symbol which occurred the majority of the time. The B complete decoder considers only those symbol estimates of $\{(\hat{s}_j, q_j)\}$ which have $q_j = 0$. A maximum likelihood decision is made on this reduced set of symbol estimates. If all symbols have $q_j = 1$, an arbitrary symbol is output. The next section describes the analysis for computing the bit error probabilities for the parallel systems using the two inner decoders described above.

III. PERFORMANCE ANALYSIS

Following the technique presented in[2], we define the following events:

$E_{s,i,0}$ = event that the first symbol out of the incomplete decoder is in error

$E_{s,i,1}$ = event that the first symbol out of the incomplete decoder is an erasure

$E_{e,i,0}$ = event that the number of errors, t, and the number of erasures, s, in the next $N-1$ symbols out of incomplete decoder satisfy $s + 2(t+1) > N-K$

$E_{e,i,1}$ = event that the number of errors, t, and the number of erasures, s, in the next $N-1$ symbols out of incomplete decoder satisfy $s + 1 + 2t > N-K$

$\widetilde{E}_{s,i}$ = event that the first symbol out of complete decoder is in error

$E_{b,i}$ = event that the first bit out of complete decoder is in error

$E_{s,o}$ = event that the first symbol out of parallel decoder is in error

$E_{b,o}$ = event that the first bit out of the parallel decoder is in error.

The overall system symbol and bit error events are given by

$$E_{s,o} = (\widetilde{E}_{s,i} \bigcap E_{s,i,0} \bigcap E_{e,i,0}) \bigcup (\widetilde{E}_{s,i} \bigcap E_{s,i,1} \bigcap E_{e,i,1}) \tag{3}$$

$$E_{b,o} = (\widetilde{E}_{b,i} \bigcap E_{s,i,0} \bigcap E_{e,i,0}) \bigcup (\widetilde{E}_{b,i} \bigcap E_{s,i,1} \bigcap E_{e,i,1}) \tag{4}$$

Noting that $E_{s,i,0}$ and $E_{s,i,1}$ are disjoint, with orthogonal signaling and frequency-hopping to insure a memoryless symbol channel we have

$$P(E_{s,o}) = P(\widetilde{E}_{s,i} \bigcap E_{s,i,0}) P(E_{e,i,0}) + P(\widetilde{E}_{s,i} \bigcap E_{s,i,1}) P(E_{e,i,1}) \tag{5}$$

$$P(E_{b,o}) = P(\widetilde{E}_{b,i} \bigcap E_{s,i,0}) P(E_{e,i,0}) + P(\widetilde{E}_{b,i} \bigcap E_{s,i,1}) P(E_{e,i,1}) \tag{6}$$

Equations (5) and (6) may be further reduced to yield:

$$P(E_{s,o}) = P(E_{s,i,0}) P(E_{e,i,0}) + P(\widetilde{E}_{s,i} \bigcap E_{s,i,1}) P(E_{e,i,1}) \tag{7}$$

$$P(E_{b,o}) = [M/\, 2(M-1)] P(E_{s,o}) \tag{8}$$

where in (7) we use the fact that

$$P(\widetilde{E}_{s,i} \mid E_{s,i,0}) = 1$$

and in (8) we applied the classical relationship of bit and symbol errors with orthogonal signaling.

To evaluate the remaining terms we need to first consider the transition probabilities of receiving a particular $(\hat{s}, q)$ pair for a given transmitted symbol. Using the notation of [1] we let

$$F_C(\theta) = Pr\{\hat{s}=i, q=0 \mid i \text{ transmitted}\}$$

$$\Delta F_C(\theta) = Pr\{\hat{s}=i, q=1 \mid i \text{ transmitted}\}$$

$$F_E(\theta) = Pr\{\hat{s}=j, q=0 \mid i \text{ transmitted}\}$$

$$\Delta F_E(\theta) = Pr\{\hat{s}=j, q=1 \mid i \text{ transmitted}\} \qquad (i \neq j)$$

It may be shown that

$$F_C(\theta) = 1 + \rho \sum_{k=1}^{M-1} \binom{M-1}{k} (-1)^k \frac{\theta}{k+\theta} \exp\left[\frac{-\rho E_s}{N_J} \frac{k}{k+\theta} \right] \tag{9}$$

$$F_E(\theta) = \frac{\rho}{M-1} \sum_{k=1}^{M-1} \binom{M-1}{k} (-1)^{k+1} \frac{\theta}{k+\theta} \frac{k}{k+\theta-1} \exp\left[\frac{-\rho E_s}{N_J} \frac{k+\theta-1}{k+\theta} \right] \tag{10}$$

$$\Delta F_C(\theta) = F_C(1) - F_C(\theta) \tag{11}$$

$$\Delta F_E(\theta) = F_E(1) - F_E(\theta) \tag{12}$$

Now define the following probabilities:

P_C = probability of correct symbol out of incomplete decoder
P_E = probability of erroneous symbol out of incomplete decoder
P_X = probability of an erasure out of incomplete decoder.

Also let I be the number of correct symbols received with good quality, J be the number of incorrect symbols received with good quality, and K be the number of correct symbols received with bad quality then the probability that the incomplete decoder is correct is given by

$$P\{I+K \geq \lfloor L/2 \rfloor+1,\ I+J \geq L-\theta_q\}.$$

Thus for decoding algorithm A

$$P_C = \sum_{i=0}^{L} \sum_{j=T1}^{L-i} \sum_{k=T2}^{L-i-j} \binom{L}{i\ j\ k} F_C^i [(M-1)F_E]^j \, \Delta F_C^k [(M-1)\Delta F_E]^{L-i-j-k} \tag{13}$$

where $T1=\max(0,\ L-\theta_q-i)$ and $T2=\max(0,\ \lfloor L/2 \rfloor+1-i)$. Similarly, we can calculate P_E for decoder A using (13) by making the following replacements

$$F_C \rightarrow F_E, \quad \Delta F_C \rightarrow \Delta F_E \text{ and } [(M-1)F_E] \rightarrow [F_C+(M-2)F_E]$$

$$[(M-1)\Delta F_E] \rightarrow [\Delta F_C+(M-2)\Delta F_E].$$

and multiplying by $M-1$. Finally it follows that

$$P_X = 1 - P_C - P_E \tag{14}$$

For decoding algorithm B it may be shown that the incomplete decoder has

$$P_C = \sum_{i=\lceil L/2 \rceil}^{L} \binom{L}{i} F_C^i [1 - F_C]^{L-i} \tag{15}$$

$$P_E = (M-1) \sum_{i=\lceil L/2 \rceil}^{L} \binom{L}{i} F_E^i [1 - F_E]^{L-i} \tag{16}$$

$$P_X = 1 - P_C - P_E \tag{17}$$

Computation of $P(E_{e,i,0})$ and $P(E_{e,i,1})$ follows directly from the definitions. These probabilities are given by

$$P(E_{e,i,0}) = \sum_{s+2t+2>N-K} \binom{N-1}{s\ t} P_X^s P_E^t P_C^{N-1-s-t} \tag{18}$$

$$P(E_{e,i,1}) = \sum_{s+2t+1>N-K} \binom{N-1}{s\ t} P_X^s P_E^t P_C^{N-1-s-t} \tag{19}$$

where for P_C, P_E, and P_X we use the appropriate values for either algorithm A or B. Noting that $P(\underset{\sim}{E}_{s,i,0}) = P_E$, and referring to equation (7), the remaining term to be evaluated is $P(\widetilde{E}_{s,i} \cap E_{s,i,1})$. This term is the probability of the joint event that the complete decoder errors and the incomplete decoder erases for a received L-tuple. Consider case A first. Let E_d be the event that no symbol of $\{\hat{s}_j;\ j=1, \ldots, L\}$ occurs more than $\lfloor L/2 \rfloor$ times. We can write:

$$P(\widetilde{E}_{s,i} \cap E_{s,i,1}) = P(\widetilde{E}_{s,i} \cap E_{s,i,1} \cap E_d) + P(\widetilde{E}_{s,i} \cap E_{s,i,1} \cap E_d^c) \tag{20}$$

$$= P(\widetilde{E}_{s,i} \cap E_d) + P(\widetilde{E}_{s,i} \cap E_{s,i,1} \cap E_d^c) \tag{21}$$

The first term of (20) is the probability that the complete decoder errors and no received symbol occurs more than $\lfloor L/2 \rfloor$ times. It can be shown that

$$P(\widetilde{E}_{s,i} \cap E_d) = \sum_{n_c=0}^{\lfloor L/2 \rfloor} \sum_{Y} G(Y, n_c) \binom{L}{Y} [F_C(1)]^{n_c} [F_E(1)]^{L-n_c} \tag{22}$$

where

n_c = the number of correct symbols received

$Y = (m_1, m_2, \ldots, m_\alpha)$

= a generic configuration of the $L - n_c$ incorrect symbols such that:

$m_1 \geq m_2 \geq \cdots \geq m_\alpha > 0$

$m_1 + m_2 + \cdots + m_\alpha = L - n_c$

$m_1 \geq n_c$

(example: $L=5$, $M=4$, the symbol set $= \{a, b, c, d\}$, if a is sent and $n_c=2$, the configuration (2,1) accounts for the received L-tuples *aabbc*, *aabbd*, *aaccd*, *aabcc*, *aabdd* and *aacdd*, the configuration (1,1,1) accounts for the received L-tuple *aabcd*)

Let, n_1 = the number of terms in the largest grouping of equal terms in Y

n_2 = the number of terms in the next largest grouping of equal terms in Y

. . .

n_β = number of terms in the smallest grouping of equal terms in Y

note: $n_1 + n_2 + \cdots + n_\beta = \alpha$

(example: (3,2) has $n_1=1$, $n_2=1$, $\beta=2$; (1,1,1) has $n_1=3$, $\beta=1$) Finally, if we define: N_c = the number of terms in Y equal to n_c and let:

$$r(N_c) = \begin{cases} 1 & \text{if } N_c = 0 \\ N_c / (N_c+1) & \text{if } N_c \neq 0 \end{cases}$$

then,

$$G(Y, n_c) = r(N_c) \frac{(M-1)(M-2)\cdots(M-\alpha)}{n_1! n_2! \cdots n_\beta!} \tag{23}$$

The term $\binom{L}{Y}$ is a shorthand notation for the multinomial coefficient $(m_1 m_2^L \cdots m_\alpha)$.

The expression for the second term of (21) is given by

$$P(\widetilde{E}_{s,i} \cap E_{s,i,1} \cap E_d^c) = (M-1) \sum_{i=0}^{L} \sum_{j=T3}^{L-i} \sum_{k=T4}^{L-i-j} \binom{L}{i\ j\ k}$$

$$\cdot F_E^i [\Delta F_E]^j [\Delta F_C + (M-2)\Delta F_E]^k [F_C + (M-2)\Delta F_E]^{L-i-j-k}$$

where $T3 = \max(0, \lfloor L/2 \rfloor + 1 - i)$, and $T4 = \max(0, \theta_q - j)$.

Calculation of $P(\widetilde{E}_{s,i} \cap E_{s,i,1})$ for algorithm B proceeds similarly.

$$P(\widetilde{E}_{s,i} \cap E_{s,i,1}) = \frac{M-1}{M} F_X^L + \sum_{l=1}^{L} \sum_{n_c=0}^{\min\{l, \lfloor L/2 \rfloor\}} \sum_{Y'} G(Y', n_c) \binom{L}{Y'} F_C^{n_c} F_E^{1-n_c} F_X^{L-1}$$

where

$$F_C = F_C(\theta), \quad F_E = F_E(\theta), \quad F_X = 1 - F_C(\theta) + (M-1)F_E(\theta)$$

l = number of terms with quality bit 0 and Y' is the same as Y in (23) except that $\sum m_i = 1 - n_c$.

IV. NUMERICAL RESULTS AND DISCUSSION

The parallel decoding system was evaluated for the case of a (32,16) Reed-Solomon code, a bit error rate (BER) of 10^{-4}, and a partial-band Gaussian noise jammer. Results for diversities $L=3$ and $L=7$ are presented in Figures 2-5. Also shown in these figures are the respective $L=3$ and $L=7$ tight upper bounds for a similar parallel decoding algorithm which requires perfect side information [2]. The bit energy-to-jammer noise ratio E_b/N_J is calculated from the symbol signal-to-jammer ratio

$$E_b/N_J = \left(\frac{N\ L}{K\ \log_2 M}\right) E_s/N_J$$

The results for algorithm A are shown in figures 2 and 3. For both cases we note that the decoder performs best when $\theta_q = L$. For this case, the incomplete decoder does not use the quality bit information and the ratio threshold test is unnecessary. At this time, the results have not been optimized for choice of θ and we hesitate to make a generalization. The algorithm B parallel decoder results, shown in figures 4 and 5, are uniformly inferior to the respective algorithm A decoders.

A good basis for comparison of the decoder performances is given in [2]. Let ρ^* be the minimum effective value of ρ. That is, for $\rho<\rho^*$ the specified BER is achieved independent of the jammer power. Similarly, let ENR^* be the maximum E_b/N_J over $0 \leq \rho \leq 1$ which will guarantee the required BER in partial-band Gaussian noise. Values of ρ^* and ENR^* for the cases shown in figures 2-5 are given in Table 1.

Table 1. Values of ρ^* and ENR^* for (32,16) R-S Code at BER 10^{-4}.

Algorithm	L	θ	θ_q	ρ^*	ENR^*
A	3	1.5	3	.33	8.26
A	3	1.5	2	.33	8.38
A	3	2.0	2	.33	8.85
A	7	1.5	7	.42	10.76
A	7	1.5	6	.42	10.77
A	7	2.0	6	.42	10.97
B	3	1.25	-	.23	10.17
B	3	1.75	-	.32	10.56
B	3	2.25	-	.34	11.66
B	7	1.5	-	.29	13.37
B	7	1.75	-	.36	13.42
B	7	2.25	-	.41	14.37
s.i.	3	-	-	.62	6.80
s.i.	7	-	-	.82	7.82

(s.i.=side information bound)

In terms of both ρ^* and ENR^* the algorithm A decoders are superior to the algorithm B decoders. Although the ρ^* values for the best algorithm A decoders shown are less than those for the case of perfect side information the ENR^* values are within 3 dB.

It should be clear that there are many different approaches that can be taken with regard to declaring erasures in Reed-Solomon codes when combined with diversity and many ways in which the algorithms presented here can be improved for better performance. For example we have not yet considered the possibility of having two Reed-Solomon decoders in parallel (without side information) as was considered in [2] with perfect side information. We have also not yet considered the performance when the M signals are implemented as

M orthogonal tones against a tone jammer or a multi-level jammer. For such an approach to be robust we would have to evaluate the performance under these situations. Of all jamming strategies for which the interference is modeled as a mixture of Gaussian random variables the worst-case strategy is two level jamming (i.e. the jammer choses two Gaussian random variables) when hard decisions are made. However for decoding strategies that employ thresholds to determine erasures this may no longer be true. Another algorithm (algorithm C) we are presently analyzing does not use any threshold on the energy detector outputs to declare erasures but just declares erasures based on $\{ s_i : i = 1,\ldots,L \}$. We believe for this algorithm that among the same class of Gaussian mixture jammers, a jammer with just two power levels is the worst in terms of minimizing decoded bit error probability. We also believe that this approach, combined with multiple Reed-Solomon decoding will be a robust decoding algorithm that will perform close to the performance when perfect side information is available. The performance analysis for decoding algorithm C will be given in a subsequent paper.

ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation under grant ECS-8451266.

REFERENCES

[1] A. J. Viterbi, "A robust ratio threshold technique to mitigate tone and partial-band jamming in coded MFSK systems," *Proceedings of the 1982 IEEE Military Communications Conference*, pp. 22.4.1-5, October, 1982.

[2] M. B. Pursley and W. E. Stark, "Performance of Reed-Solomon coded frequency-hop spread spectrum in partial-band interference," *IEEE Transactions on Communications*, pp. 767-774, August, 1985.

[3] Pil Joong Lee, "Performance of a normalized energy metric without jammer state information for an FH/MFSK system in worst case partial band jamming," *IEEE Transaction on Communications*, vol. COM-33, pp. 869-877, August, 1985.

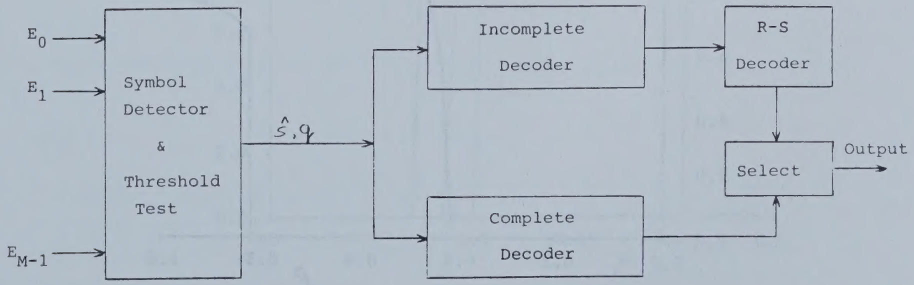


Figure 1. Parallel Decoder.

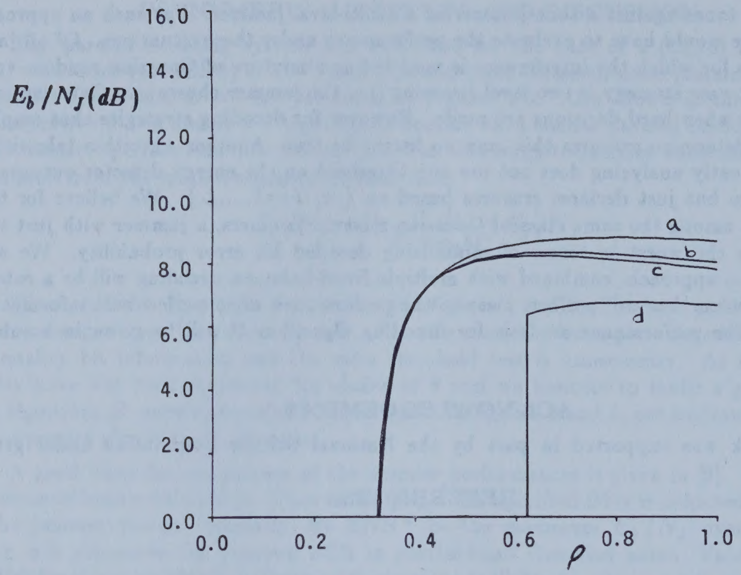


Figure 2. E_b/N_J necesary for 10^{-4} BER, Algorithm A, diversity 3. (a) $\theta=2.0$, $\theta_q=2$; (b) $\theta=1.5$, $\theta_q=2$; (c) $\theta=1.5$, $\theta_q=3$; (d) side information bound.

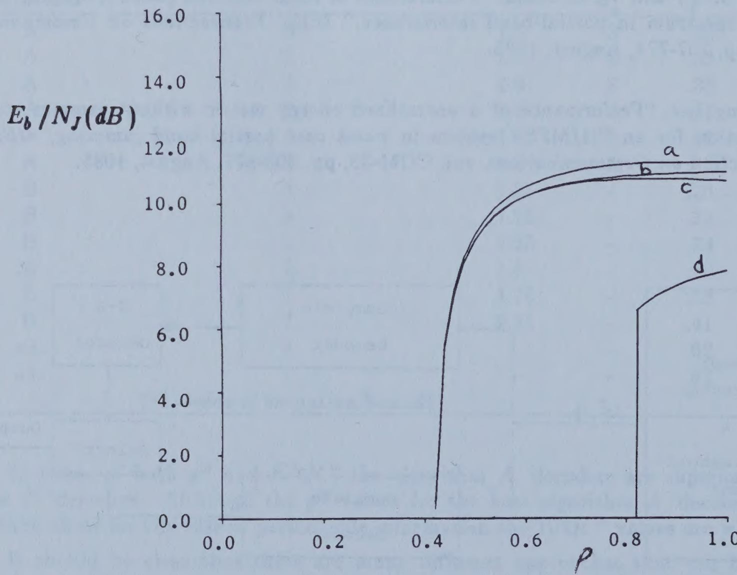


Figure 3. E_b/N_J necesary for 10^{-4} BER, Algorithm A, diversity 7. (a) $\theta=1.5$, $\theta_q=4$; (b) $\theta=2.0$, $\theta_q=6$; (c) $\theta=1.5$, $\theta_q=7$; (d) side information bound.

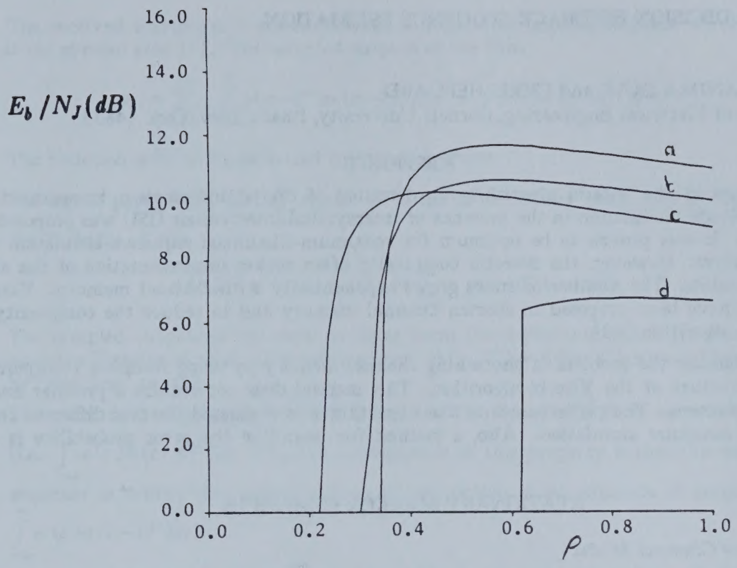


Figure 4. E_b/N_J necesary for 10^{-4} BER, Algorithm B, diversity 3. (a) θ=2.25; (a) θ=1.75; (a) θ=1.25; (d) side information bound.

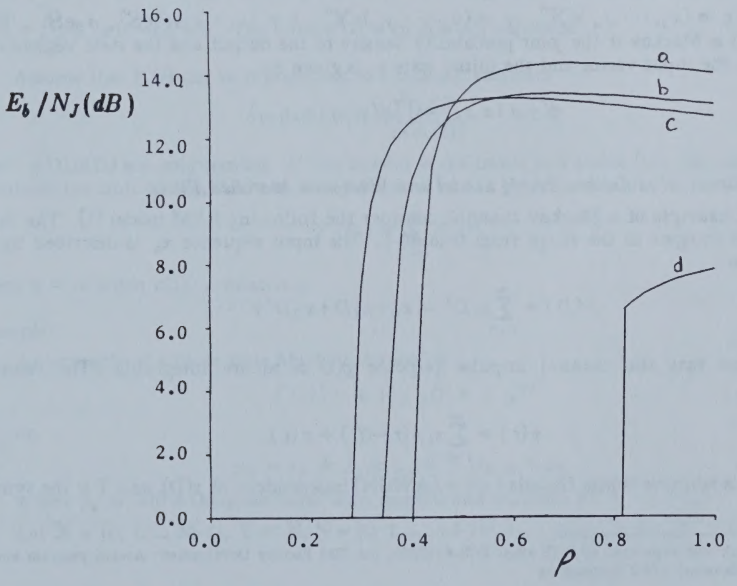


Figure 5. E_b/N_J necesary for 10^{-4} BER, Algorithm B, diversity 7. (a) θ=2.25; (a) θ=1.75; (a) θ=1.50; (d) side information bound.

DELAYED DECISION FEEDBACK SEQUENCE ESTIMATION.

ALEXANDRA DUEL and CHRIS HEEGARD
School of Electrical Engineering, Cornell University, Ithaca, New York 14853

ABSTRACT

The use of the Viterbi algorithm for detection of digital information transmitted by pulse amplitude modulation in the presence of intersymbol interference (ISI) was proposed by Forney [1]. It was proven to be optimum for maximum-likelihood sequence estimation and easy to analyze. However, the detector complexity often makes implementation of the algorithm impossible. The number of states grows exponentially with channel memory. Various approaches have been proposed to shorten channel memory and to reduce the complexity of the Viterbi algorithm [2,3].

We consider the problem of shortening channel memory by using feedback incorporated into the structure of the Viterbi algorithm. This method does not require a prefilter and is easy to implement. The performance of the algorithm was evaluated for two different channels using computer simulation. Also, a method for bounding the error probability is discussed.

STATIONARY MARKOV CHANNELS

The Markov Channel Model

Consider the following discrete time channel model. The input to the channel is a sequence of symbols drawn from a finite alphabet $\mathbf{X}$. The output symbols lie in space $\mathbf{Y}$. The channel is described by its state space $\mathbf{S}$ and a conditional probability density function ϕ. For each $x \in \mathbf{X}$, $s, s^- \in \mathbf{S}$, $y \in \mathbf{Y}$, $\phi_{\mathbf{Y},\mathbf{S}|\mathbf{X},\mathbf{S}}(y,s \mid x,s^-)$ describes the distribution of next state and output given the previous state and input.

Let $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbf{X}^n$, $\boldsymbol{y} = (y_1, \cdots, y_n) \in \mathbf{Y}^n$, $\boldsymbol{s} = (s_1, \cdots, s_n) \in \mathbf{S}^n$, $s_0 \in \mathbf{S}$. Then the channel is Markov if the joint probability density of the output and the state vectors conditioned on the input vector and the initial state s_0 is given by

$$\phi(\boldsymbol{y},\boldsymbol{s} \mid \boldsymbol{x},s_0) = \prod_{i=1}^{n} \phi(y_i,s_i \mid x_i,s_{i-1}).$$

Pulse Amplitude Modulation (PAM) Model and Whitened Matched Filter

As an example of a Markov channel, consider the following PAM model [1]. The input symbols are integers in the range from 0 to M-1. The input sequence x_n is described by it's D-transform

$$x(D) = \sum_{i=0}^{\infty} x_i D^i = x_0 + x_1 D + x_2 D^2 + \cdots$$

Assume that the channel impulse response p(t) is square integrable. The received waveform

$$y(t) = \sum_{i=0}^{\infty} x_i \, p(t-iT) + z(t),$$

where z(t) is additive white Gaussian noise (AWGN) independent of x(D), and T is the symbol period.

This work was supported by NSF grant ECS-8352220, the IBM Faculty Development Award program and an AT&T Bell Laboratories PhD Scholarship.

The received waveform is passed through a filter with impulse response w(-t) and sampled at the symbol rate 1/T. The sampled outputs of the filter

$$y_k = \sum_{i=0}^{\infty} x_i \int_{-\infty}^{\infty} p(\tau - iT) w(\tau - kT) d\tau + \int_{-\infty}^{\infty} z(\tau) w(\tau - kT) d\tau .$$

The function w(t) is characterized by two properties:

(i) The signal part of y(t) (i.e., $\sum_{i=0}^{\infty} x_i p(t-iT)$) lies in the vector space $A(p,T)$ spanned by the *T-shifts* of p(t)

$$A(p,T) = \{s(t) \mid s(t) = \sum_{i=0}^{N} \alpha_i p(t-iT),\ \alpha_i \in \mathbf{R}\}$$

The sampled outputs of the receiver filter form the coefficients of the projection of y(t) onto the T-shifts of w(t). It is required that $A(p,T) = A(w,T)$; such a filter is said to be "matched" to p(t).

(ii) The other desired property for w(t) is that the T-shifts of w(t) form an orthonormal set (i.e., $\int_{-\infty}^{\infty} w(t) w(t-kT) dt = \delta_k$). A consequence of this property is that the output noise sequence is white; the autocorrelation of the output noise sequence is proportional to $\int_{-\infty}^{\infty} w(t) w(t-kT) dt$.

A receiver filter which satisfies both (i) and (ii) is termed a *whitened match filter.*

The sampled outputs of the whitened matched filter satisfy

$$y(D) = f(D) x(D) + z(D),$$

where $f(D) f(D^{-1}) = R_p(D)$, the sampled pulse autocorrelation function of p(t) defined by $(R_p)_k = \int_{-\infty}^{\infty} p(t) p(t-kT) dt$. The noise z(D) is an AWGN sequence.

Assume that f(D) can be represented as a rational function

$$f(D) = \frac{g(D)}{h(D)}, \tag{1}$$

where g(D),h(D) are polynomials. If the system is realizable and stable (i.e., the roots of h(D) lie outside the unit circle), f(D) can be represented by the power series

$$f(D) = \sum_{i=0}^{\eta} f_i D^i,$$

where $\eta = \infty$ when h(D) is nontrivial.

Examples

1. An example of a finite state Markov channel is

$$f(D) = 1 + f_1 D + f_2 D^2$$

or

$$y_k = x_k + f_1 x_{k-1} + f_2 x_{k-2} + z_k ,$$

where z_k is white Gaussian noise with mean 0 and variance σ^2.

Let $\mathbf{X} = \{0, 1, ..., M\text{-}1\}$, $\mathbf{Y} = \mathbf{R}$, $\mathbf{S} = \{0, 1, ..., M\text{-}1\}^2$, $s_{k-1} = (x_{k-1}, x_{k-2})^T$. Denote

$\theta = (f_1, f_2)^T$. Then s_k = function (x_k, s_{k-1}) and

$$y_k = x_k + \theta^T s_{k-1} + z_k.$$

Thus,

$$\phi(y_k \mid s_{k-1}, x_k) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left[-\frac{(y_k - x_k - \theta^T s_{k-1})^2}{2\sigma^2}\right].$$

2. The second example deals with an infinite state Markov channel

$$f(D) = \frac{1}{1-\alpha D}, \quad 0 < \alpha < 1. \tag{2}$$

Let

$$w(D) = \frac{x(D)}{1-\alpha D}$$

Then

$$y_k = x_k + \alpha w_{k-1} + z_k, \quad w_k = x_k + \alpha w_{k-1}$$

where z_k is white Gaussian noise with mean 0 and variance σ^2.
Let $\mathbf{X} = \{0,1, \cdots, M-1\}$, $\mathbf{Y} = \mathbf{S} = \mathbf{R}$. Set $s_{k-1} = w_{k-1}$. Then s_k = function(x_k, s_{k-1}) and

$$\phi(y_k \mid s_{k-1}, x_k) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left[-\frac{(y_k - x_k - \alpha s_{k-1})^2}{2\sigma^2}\right].$$

In the general PAM model, let f be as in (1) and

$$w(D) = f(D)\, x(D)$$

(assume $h_0 = g_0 = 1$). Then

$$y_k = x_k + \sum_{i=1}^{q} g_i x_{k-i} - \sum_{i=1}^{m} h_i w_{k-i} + z_k,$$

where z_k is white Gaussian noise with mean 0 and variance σ^2. Let $\mathbf{X} = \{0,1, \cdots, M-1\}$, $\mathbf{Y} = \mathbf{R}$, $\mathbf{S} = \mathbf{X}^q \times \mathbf{R}^m$. Set $s_{k-1} = (x_{k-1}, \cdots, x_{k-q}, w_{k-1}, \cdots, w_{k-m})^T$. Then s_k = function (x_k, s_{k-1}). Let $\theta = (g_1, \cdots, g_q, -h_1, \cdots, -h_m)^T$. Then

$$\phi(y_k \mid s_{k-1}, x_k) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left[-\frac{(y_k - x_k - \theta^T s_{k-1})^2}{2\sigma^2}\right].$$

DELAYED DECISION FEEDBACK SEQUENCE ESTIMATION (DDFSE).

Branch Metric and Partial State Estimator

Consider a Markov channel $(\mathbf{X}, \mathbf{Y}, \mathbf{S}, \phi(y, s \mid x, s^-))$, where we can factor $\mathbf{S} = \mathbf{U} \times \mathbf{V}$, where $|\mathbf{U}| < \infty$. Then

$$\phi(y, s \mid x, s^-) = \phi(y, u \mid x, u^-, v^-)\, \phi(v \mid y, u, x, u^-, v^-).$$

A *branch metric* m is in general a function from $\mathbf{Y} \times \mathbf{U} \times \mathbf{X} \times \mathbf{U} \times \mathbf{V}$ to $\mathbf{R}$. Given an input x and a previous state $s^- = (u^-, v^-)$, $m(y, u, x, u^-, v^-)$ measures the possibility of occurrence of an output y and a partial state u according to some criterion. A *maximum likelihood branch metric* is

$$m(y,u,x,u^-,v^-) \sim [-\log \phi(y,u \mid x,u^-,v^-)].$$

A *partial state estimator* is defined as a function $\hat{v}$ from $\mathbf{Y}\times\mathbf{U}\times\mathbf{X}\times\mathbf{U}\times\mathbf{V}$ to $\mathbf{V}$. It associates an estimate of the partial state v with an output y, a partial state u, an input x and a previous state (u^-,v^-). The *maximum likelihood partial state estimator* is given by $\hat{v}(y,u,x,u^-,v^-)$ which minimizes $-log\ \phi(v \mid y,u,x,u^-,v^-)$, i.e.

$$-\log \phi(\hat{v} \mid y,u,x,u^-,v^-) = \inf_{v\in V} [-\log \phi(v \mid y,u,x,u^-,v^-)].$$

Examples

Reconsider the examples for the PAM model.

1. Let $u_{k-1} = x_{k-1}$, $v_{k-1} = x_{k-2}$. Since u_k is a function of x_k, a maximum likelihood branch metric is given by

$$m(y_k,u_{k-1},v_{k-1}) = (y_k - x_k - f_1 u_{k-1} - f_2 v_{k-1})^2.$$

The maximum likelihood partial state estimator is

$$\hat{v}(u_{k-1}) = u_{k-1}.$$

2. From (2) we get

$$y(D) = x(D)(1 + \alpha D + \ldots + \alpha^{\mu} D^{\mu}) + D^{\mu+1} w(D) + z(D)$$

where

$$w(D) = \frac{\alpha^{\mu+1}}{1-\alpha D} x(D).$$

Then

$$y_k = x_k + \alpha x_{k-1} + \cdots + \alpha^{\mu} x_{k-\mu} + w_{k-\mu-1} + z_k,$$

where

$$w_k = \alpha^{\mu+1} x_k + \alpha\, w_{k-1}.$$

Define $\mathbf{U} = \mathbf{X}^{\mu}$, $\mathbf{V} = \mathbf{R}$, $u_{k-1} = (x_{k-1}, \cdots, x_{k-\mu})^T$ and $v_{k-1} = w_{k-\mu-1}$. Let $\theta = (\alpha, \cdots, \alpha^{\mu})^T$. Since u_k is a function of (x_k, u_{k-1}),

$$\phi(y_k \mid x_k, u_{k-1}, v_{k-1}) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left[-\frac{(y_k - x_k - \theta^T u_{k-1} - v_{k-1})^2}{2\sigma^2}\right].$$

Then the maximum likelihood branch metric is given by

$$m(y_k,x_k,u_{k-1},v_{k-1}) = (y_k - x_k - \theta^T u_{k-1} - v_{k-1})^2.$$

The maximum likelihood partial state estimator is

$$\hat{v}(u_{k-1},v_{k-1}) = \alpha^{\mu+1} x_{k-\mu} + \alpha v_{k-1}.$$

In general, if $f(D) = \sum_{i=0}^{\eta} f_i D^i$, define

$$f_{\mu}(D) = \sum_{i=0}^{\mu} f_i D^i, \tag{3}$$

(where $\mu \leqslant \eta$) and

$$f_+(D) \equiv D^{-(\mu+1)}(f(D) - f_\mu(D)) = \sum_{i=0}^{\eta-\mu-1} f_{i+\mu+1} D^i . \tag{4}$$

Then

$$y(D) = f_\mu(D)\, x(D) + D^{\mu+1} f_+(D)\, x(D) + z(D).$$

Note that

$$f_+(D) = D^{-(\mu+1)} (f(D) - f_\mu(D))$$

$$= D^{-(\mu+1)} \frac{g(D) - f_\mu(D)\, h(D)}{h(D)} = \frac{\hat{g}(D)}{h(D)}$$

is rational.

Thus,

$$f_+(D) = \frac{\hat{g}(D)}{h(D)} = \frac{\sum_{i=0}^{l} \hat{g}_i D^i}{\sum_{i=0}^{m} h_i D^i}.$$

Let

$$w(D) = f_+(D)\, x(D) = \frac{\hat{g}(D)}{h(D)}\, x(D)$$

or

$$w_k = x_k + \sum_{i=1}^{l} \hat{g}_i x_{k-i} - \sum_{i=1}^{m} h_i w_{k-i} .$$

Then, at time k

$$y_k = x_k + f_1 x_{k-1} + \cdots + f_\mu x_{k-\mu} + w_{k-\mu-1} + z_k.$$

Set $u_{k-1} = (x_{k-1}, \cdots, x_{k-\mu})^T$, $v_{k-1} = (x_{k-\mu-1}, \cdots, x_{k-\mu-l}, w_{k-\mu-1}, \cdots, w_{k-m-\mu})^T$. Let $\theta = (f_1, \cdots, f_\mu)^T$. Since u_k is a function of x_k and u_{k-1}, the maximum likelihood branch metric is given by

$$m(y_k, x_k, u_{k-1}, v_{k-1}) = (y_k - x_k - \theta^T u_{k-1} - w_{k-\mu-1})^2. \tag{5}$$

The maximum likelihood partial state estimator is

$$\hat{v}(u_{k-1}, v_{k-1}) = (x_{k-\mu}, \cdots, x_{k-\mu-l+1}, w_{k-\mu}, \cdots, w_{k-m+1-\mu})^T . \tag{6}$$

where

$$w_{k-\mu} = x_{k-\mu} + \sum_{i=1}^{l} \hat{g}_i x_{k-\mu-i} - \sum_{i=1}^{m} h_i w_{k-\mu-i} .$$

DDFSE Algorithm

Consider a Markov channel with state decomposition $\mathbf{S} = \mathbf{U} \times \mathbf{V}$, $|\mathbf{U}| < \infty$. Fix a branch metric m and a partial state estimator $\hat{v}$. The output of the channel is given by the sequence $(y_1, \cdots, y_N)$. Our objective is to estimate the input sequence $(x_1, \cdots, x_N)$.

Suppose at time k we associate with each $u \in U$ a metric $M_{k-1}(u)$, a partial state estimate $\tilde{v}_{k-1}(u)$ and a vector of input estimates $\tilde{x}_{k-1}(u) = (\tilde{x}_1(u), \cdots, \tilde{x}_{k-1}(u))$. The metric is recursively defined as

$$M_k(u) = \min_{x \epsilon X, u^- \epsilon U} [M_{k-1}(u^-) + m(y_k, u, x, u^-, \tilde{v}_{k-1}(u^-))].$$

Let the minimum be achieved for some (x^0, u^-). Then assign $\tilde{v}_k(u) = \hat{v}(y_k, u, x^0, u^-, \tilde{v}_{k-1}(u^-))$, and set $\tilde{x}_k(u) = (\tilde{x}_1(u^-), \cdots, \tilde{x}_{k-1}(u^-), x^0)$.

Formally, the DDFSE algorithm is described as follows:

```
For $(u \epsilon U)$ { Initialize $(M(u), \tilde{v}(u))$}

For $(k = 1, \cdots, N)$

        { Read $y_k$; For $(u \epsilon U)$ {$M_-(u) = M(u)$; $\tilde{v}^-(u) = \tilde{v}(u)$}

        For $(u \epsilon U)$

                { Let $x^0 \epsilon X, u^- \epsilon U$ minimize

                $M_-(u^-) + m(y_k, u, x^0, u^-, \tilde{v}^-(u^-))$.

                Then $M(u) = M_-(u^-) + m(y_k, u, x^0, u^-, \tilde{v}^-(u^-))$;

                        $\tilde{v}(u) = \hat{v}(y_k, u, x^0, u^-, \tilde{v}^-(u^-))$;

                        $P_k(u) = u^-$; $\tilde{x}_k(u) = x^0$;

                }

        }
```

Let $\hat{u}_n$ *minimize* $M(u)$ *then* $\hat{u}_0, \cdots, \hat{u}_{n-1}$ *are given by* $\hat{u}_{k-1} = P_k(\hat{u}_k)$

and $\hat{x}_1, \cdots, \hat{x}_n$ *are given by* $\hat{x}_k = \tilde{x}(\hat{u}_k)$.

Note that if $\mathbf{U} = \mathbf{S}$ and m is the maximum likelihood branch metric, the algorithm reduces to the Viterbi algorithm(VA). If $\mathbf{V} = \mathbf{S}$, the algorithm is equivalent to decision-feedback equalization (if m and $\hat{v}$ are maximum likelihood, the algorithm operates with a maximum likelihood criteria for detection of the input symbol and estimation of the state).

Suppose the memory constraint $\mu \leqslant \eta$ is imposed on a detection scheme for the PAM channel with $f(D) = \sum_{i=0}^{\eta} f_i D^i$. Since VA requires M^η states, it can not be used if $\mu < \eta$. Suppose an estimate of x(D) from y(D) is made by minimizing the Euclidean norm of $y(D) - f_\mu(D)x(D)$ [3], where $f_\mu(D)$ is given by (3). The remaining interference $f_+(D)$ (4) is treated as if it is equal to zero. Such a reduced state algorithm ignores remaining ISI. It has M^μ states. This is equivalent to DDFSE with partial state estimator $\hat{v} = 0$. The effective noise term in this detection scheme is $f_+(D)x(D) + z(D)$. The term $f_+(D)x(D)$ is the residual interference for this algorithm.

Suppose DDFSE is used for this channel with m and $\hat{v}$ given by (5) and (6). Then an estimate of the remaining interference $w_{k-\mu-1}$ is made at each state u using partial state estimate $\tilde{v}(u)$. The residual interference for this detection scheme is the difference between the correct and the estimated remaining interference terms. Suppose $\tilde{w}(D) = f_+(D)\tilde{x}(D)$ is the estimate of remaining interference associated with some path P in the DDFSE. Then the residual interference for this path is

$$e(D) = D^{\mu+1}(w(D) - \tilde{w}(D)).$$

The adjusted output sequence is

$$\tilde{y}(D) = f_{\mu}(D)x(D) + e(D) + z(D).$$

COMPUTER SIMULATION RESULTS AND PERFORMANCE ANALYSIS

Computer simulation was used to evaluate performance of the DDFSE for the channels given in the examples.

1. Consider the discrete time model of magnetic recording channel with colored noise [5,6]. The output of the channel is

$$y_k = x_k - x_{k-1} + \tilde{z}_k,$$

where x_k is binary and $\tilde{z}_k$ is colored Gaussian noise with autocorrelation $(R_{\tilde{z}})_k = \sigma^2 r^{|k|}$. After being passed through the whitened filter, the output of the channel is given by the following transfer function

$$f(D) = (1 - D)(1 - rD), \quad |r| < 1. \tag{7}$$

This is example 1 with $f_1 = -(1+r)$, $f_2 = r$ and M=2. The DDFSE algorithm allows reduction from four to two states. Computer simulation for $r = ½$ shows that DDFSE has bit error rate close to that of the VA with four states, and is much better in it's performance than reduced state algorithm without feedback (see figure 1).

The following upper bound on the bit error rate of DDFSE is shown in figure 1.

Lemma. For the channel (7) with $r = ½$, an upper bound on bit error rate of DDFSE with $\mu = 1$ is given by

$$P_b(E) < \sum_{n=1}^{\infty} 2^n \left[\sum_{i_1, \cdots, i_n = 1}^{\infty} \frac{i_1 + \cdots + i_n}{2^{i_1 + \cdots i_n}} \right] Q\left((1/2)\,SNR^{½}\right)\left(Q\left((5/18)\,SNR^{½}\right)\right)^{n-1}$$

$$= \frac{4Q\left((1/2)\,SNR^{½}\right)}{\left(1 - 2\,Q\left((5/18)\,SNR^{½}\right)\right)^2}, \tag{8}$$

where SNR is the ratio of peak signal energy to noise energy at the output of the whitened matched filter.

This bound is obtained by the following argument.

Let $f(D) = 1 + f_1 D + f_2 D^2$, $\mu = 1$. Suppose $a(D)$ and $\hat{a}(D)$ are correct and estimated signal sequences between times k_1 and k_2. Assume that the paths associated with $a(D)$ and $\hat{a}(D)$ start at the same state u_1 and terminate at the same state u_2. The received output sequence is $y(D) = a(D) + z(D)$, where $z(D)$ denotes noise in the given time interval. Let $\rho(D)$ be the residual interference for $y(D)$ associated with the path leading to the state u_1 at time k_1. Then the adjusted output sequence is $\tilde{y}(D) = y(D) + \rho(D)$ (see figure 2). Then at time k_2 the path leading to u_2 is chosen incorrectly if

$$||\tilde{y}(D) - a(D)|| > ||\tilde{y}(D) - \hat{a}(D)||.$$

However, only the projection of $\rho(D) + z(D)$ on the line joining $a(D)$ and $\hat{a}(D)$ matters in the decision process. Let ρ_0 be the length of the projection of $\rho(D)$ on $a(D) - \hat{a}(D)$. Let d be the distance between $a(D)$ and $\hat{a}(D)$. Then the probability that the incorrect path is chosen is upper bounded by [2]

$$P(e) \leqslant C_1 \max_{d,\rho_0} Q\left(\left(\frac{d}{2} - \rho_0\right) / \sigma_z\right), \tag{9}$$

where σ_z^2 is the noise variance. Note that $\rho_0 = f_2 / d$ if $\rho(D) \neq 0$. Suppose that the minimum in (9) is achieved for some $d = d_{\min}$. For the channel (7) with $r = ½$, $d_{\min} = (3/2)A$, where A is the amplitude of the input symbols. The Q-function in the numerator of (8) is obtained from (9) when $\rho_0 = 0$. The Q-function in the denominator of (8) corresponds to (9) when $\rho_0 \neq 0$. The upper bound on the bit error rate (8) is a union bound. It is the sum over all most probable independent sequences of erroneous events described above. The first event has $\rho_0=0$ and the rest have $\rho_0 \neq 0$. In the sum, the probability of occurrence of each sequence is weighed by the number of error bits it entails.

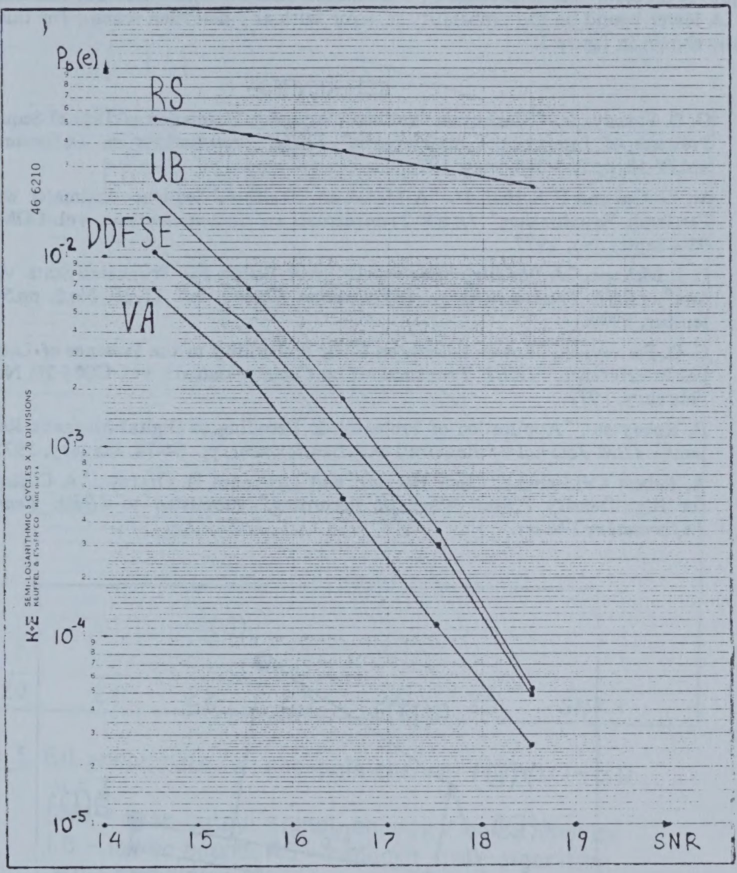


Figure 1 Bit error rate for the channel $f(D) = (1-rD)(1-D)$.
$\mu = 1$, $r = 1/2$.
SNR = peak output signal energy / noise energy
UB - upper bound for DDFSE,
RS - reduced state algorithm with remaining ISI ignored , VA - Viterbi Algorithm

2. The second channel has continuous time impulse response $h(t) = e^{-t/\tau}, t \geqslant 0$. The corresponding discrete time response is $f(D) = \sum_{i=0}^{\infty} \alpha^i D^i$ (2), where $\alpha = e^{-T/\tau}$. Since $\eta = \infty$, Viterbi algorithm can not be applied to this channel. DDFSE for this channel significantly improves bit error rate achievable by the reduced state algorithm without feedback (see figure 3). A lower bound on the probability of error with any detection scheme for this channel [4] is also shown in figure 3.

REFERENCES

1. G. D. Forney, Jr., "Maximum-likelihood Sequence Estimation of Digital Sequences in the Presence of Intersymbol Interference," *IEEE Transactions on Information Theory*, vol.IT-18, pp.363-378, May, 1972.
2. W. U. Lee and F. S. Hill, Jr., "A Maximum-likelihood Sequence Estimator with Decision-Feedback Equalization," *IEEE Transactions on Communications*, vol. COM-25, pp.971-980, September, 1977.
3. P. J. McLane, "A Residual Interference Error Bound for Truncated-State Viterbi Detectors," *IEEE Transactions on Information Theory*, vol. IT-26, No.5, pp.548-553, September, 1980.
4. G. D. Forney, Jr., "Lower Bounds on Error Probability in the Presence of Large Intersymbol Interference," *IEEE Transactions on Communications*, vol. COM-20, No.1, pp.76-77, February, 1972.
5. H. Kobayashi, "Application of Probabilistic Decoding to Digital Magnetic Recording Systems," *IBM Journal of Research and Development*, pp. 64-74, January, 1971.
6. A. Robert Calderbank, Chris Heegard and Lawrence H. Ozarow, "A Coding Technique for High-Density Digital Magnetic Recording," submitted to *IEEE Transactions on Information Theory*.

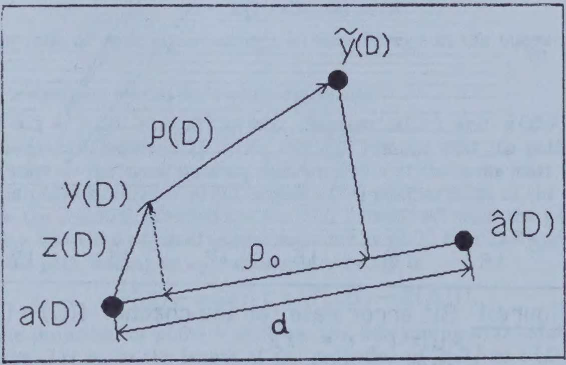


Figure 2. Decision process in the signal space.

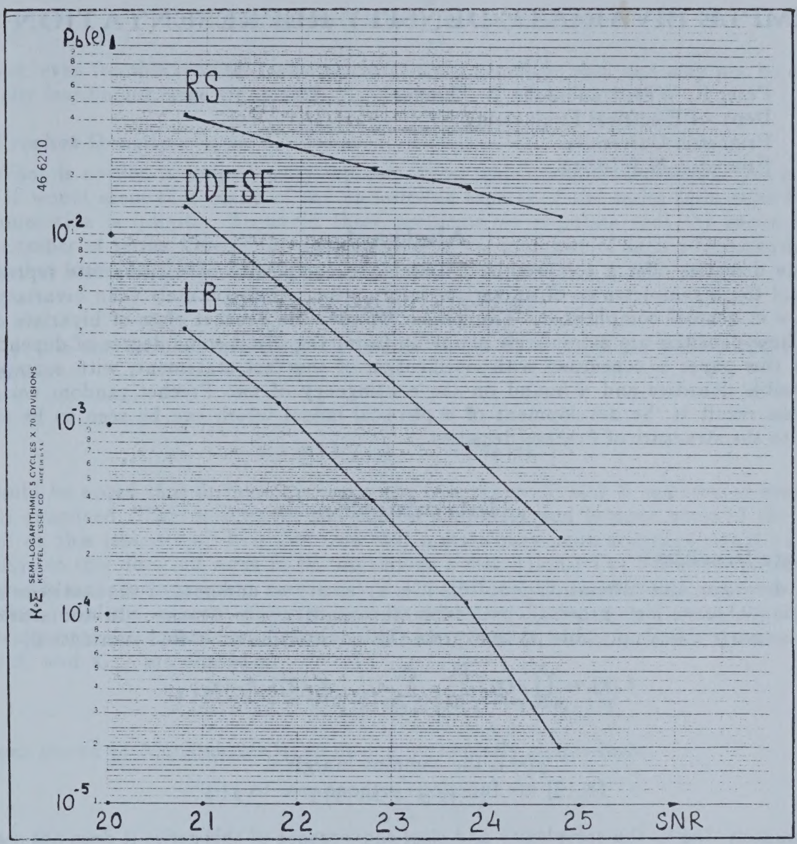


Figure 3. Bit error rate for one-pole channel $f(D)=1/1-\alpha D$.
$\mu = 3$.
SNR = peak output signal energy / noise energy
LB - lower bound. RS - reduced state algorithm
with remaining ISI ignored.

A SIMPLE BIVARIATE DENSITY REPRESENTATION

Peter K. Willett and John B. Thomas
Dept. of Electrical Engineering
Princeton University
Princeton N.J. 08544

Abstract

Bivariate densities offer a compromise between the complexity of multivariate representations and the incompleteness of univariate information. Unfortunately even bivariate densities are in general complicated. This paper explores the Frechet class of bivariate densities. These densities are attractively simple in form, yet admit some degree of dependence. Part of this paper is concerned with clarification of this last statement, with examples of inadmissible densities and a bound on the contingency of two Frechet random variables. The main result is the development of a physical model which can be seen to be closely related to the structure of Frechet densities.

Introduction

Bivariate Densities

In detection and estimation problems it is necessary to understand the statistics of the underlying observations process, preferably the multivariate density. In the stationary case it is nearly always possible [1] to express this quantity as an n-fold summation,

$$f(\bar{\mathbf{x}}) = \prod_{i=1}^{n} g(x_i) \sum_{j_1=0}^{\infty} \cdots \sum_{j_n=0}^{\infty} c_{j_1 \ldots j_n} \theta_{j_1}(x_1) \ldots \theta_{j_n}(x_n)$$

where

$g(\cdot)$ is the common marginal
$\{\theta_k(\cdot)\}$ are functions orthonormal to $g(\cdot)$

Unfortunately due to the complexity and slow convergence of this series it does not seem to have much more than theoretical interest.

There are more tractable models in some cases. It is often assumed that the density is Gaussian:

$$f(\bar{\mathbf{x}}) = \left[(2\pi)^n \mid \Lambda \mid\right]^{-\frac{1}{2}} \exp(-\tfrac{1}{2}\bar{\mathbf{x}}^T \Lambda^{-1} \bar{\mathbf{x}})$$

where Λ is the correlation matrix of the process An alternative assumption often made is that the process is stationary with independent samples, so that

$$f(\bar{\mathbf{x}}) = \prod_{i=1}^{n} g(x_i)$$

where $g(\cdot)$ is the common marginal.

A third possibility is to assume that the process can be described sufficiently by its bivariate densities. This is useful, for example, if the process is Markov, so that

$$f(\bar{\mathbf{x}}) = f(x_1) \prod_{i=2}^{n} f(x_i \mid x_{i-1})$$

or if higher-order information is unnecessary, as with the calculation of the efficacy of a memoryless statistic. It is on this assumption that the paper will proceed.

A problem is that even in the bivariate case the representation of dependent non-Gaussian densities is in general intractable. Lancaster [2] showed that most bivariate densities with common marginals can be expressed as the following *canonical* expansion:

$$f(x,y) = g(x)g(y)\sum_{i=0}^{\infty} c_i\,\theta_i(x)\eta_i(y)$$

However, even for this representation convergence is generally slow and may not be computationally feasible for large arguments.

The Frechet Density

What is needed is some density representation which admits dependence yet is tractable. It would seem that many of the undesirable aspects of the series form exist because the summation is infinite. To avoid these problems we stipulate that the series have a finite number of terms. One of the simplest non-trivial examples of such a representation is the Frechet form [3],[4],[5]:

$$f(x,y) = g(x)g(y)\left[1+\gamma\theta(x)\theta(y)\right]$$

where $g(\cdot)$ is the common marginal. It is easy to see that there are some restrictions on $\theta(\cdot)$; namely

$$\int g(z)\theta(z)\,dz = 0$$

$$\gamma\theta(x)\theta(y) \geq -1 \qquad (\forall x,y)$$

(It should be noted that in this paper only the case where X and Y are treated symmetrically is examined. This is done for the sake of simplicity and because some of the results depend on this restriction. It is also true that a stationary noise sequence would have this property, so this does not seem to be too serious a restriction.) The Frechet representation is very advantageous in some senses; it is straightforward to calculate, and one avoids numerical problems inherent in series, such as wildly oscillating partial sums. In addition, the locally-optimal memoryless detector $T(\mathbf{\overline{x}})$ for a known signal in Frechet noise, that is where X_i and X_{i+j} are related as

$$f(x_i,x_{i+j}) = g(x_i)g(x_{i+j})\left(1+\gamma_j\,\theta(x_i)\theta(x_{i+j})\right)$$

has been shown by Moustakides [4] to have a particularly simple form:

$$T(\mathbf{\overline{x}}) = \sum_{i=1}^{n} s_i\,\Psi(x_i)$$

$$\Psi(z) = \frac{-g'(z)}{g(z)} - c\,\theta(z)$$

where

s_i is the i^{th} signal
c is a constant dependent on $\{s_i\}$ and $\{\gamma\}$

Applicability of Frechet Densities

The question naturally arises as to how general a density this representation admits. Moustakides has given some bounds on the correlation ρ of Frechet random variables. For example, he showed that if the marginals are Gaussian, then

$$|\rho| < .64$$

with similar results for other marginals.

Degeneracy of Frechet Densities

The above is an interesting result, yet it is only the partial story. Indeed, it can be shown that no bivariate Gaussian density with non-zero correlation can be represented as a Frechet density. Actually, such a statement can be made much more widely with reference to Lancaster's canonical series . It has been shown [2] that almost every bivariate density can be expanded into a single summation of orthonormal functions, and that the requisite choice of these functions is unique. Since the Frechet representation has only a single summation, it would appear that the Frechet series *is* the canonical series (see Appendix A). This means that the Frechet density is in a sense degenerate, and furthermore that any non-degenerate bivariate density --a density with a full orthonormal expansion-- cannot be Frechet. That is, if we can write

$$f(x,y) = g(x)g(y)\left[1 + \sum_{i=1}^{\infty} c_i \eta_i(x)\eta_i(y)\right]$$

where

$g(\cdot)$ is the common marginal
$\{\eta_i(\cdot)\}$ are functions orthonormal to $g(\cdot)$

and $c_i \neq 0$ for more than one value of i, then $f(x,y)$ cannot be Frechet. For example, in the bivariate Gaussian case, the η 's are Hermite polynomials and $c_i = \rho^i$

A Bound on the Contingency

Another description of those densities in the Frechet class involves Pearson's contingency, usually denoted ϕ^2. Unlike the correlation, which seems to have global meaning only in special cases like the Gaussian case, this quantity measures the dependency between two general random variables. The contingency ϕ^2 is defined as

$$\phi^2 = \int\int \frac{[f(x,y)]^2}{g(x)g(y)} dxdy - 1$$

and can be related to the expected value of the mutual information of X and Y. That is, we can write

$$\phi^2+1 = \int\int \left[\frac{f(x,y)}{g(x)g(y)}\right] f(x,y)dxdy$$

or

$$\phi^2+1 = \mathbf{E}\left\{ e^{I(X,Y)} \right\}$$

where

$I(X,Y)$ is the mutual information between X and Y

It is known that

$$\phi^2 = 0 \iff X \text{ and } Y \text{ are independent}$$
$$\phi^2 = \infty \iff X \text{ and } Y \text{ are functionally related}$$

It can be shown (see Appendix B) that for a density in the Frechet class

$$\phi^2 \leq 1$$

In general it is true that $\phi^2 \in (0,\infty)$ which implies that $0 \leq \phi^2 \leq 1$ is a severe restriction on Frechet densities. However, it should be borne in mind that in the bivariate Gaussian case

$$\phi^2 \leq 1 \iff |\rho| \leq \frac{1}{\sqrt{2}} \simeq 0.7$$

which is a relatively lax bound. (Note that this should not be compared to Moustakides' bound; no non-trivial Gaussian density is in the Frechet class.)

A Physical Model for Frechet Densities

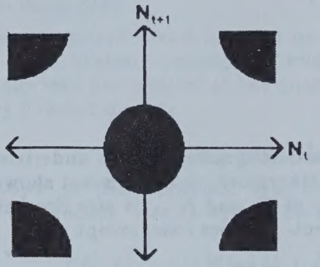
The previous comments have centered on the applicability of the Frechet density representation to a noise sequence with given statistics. Perhaps a better understanding may be gained from a physical model which produces a Frechet bivariate density.

The Model and its Bivariate Density

In order to do this, let us construct a Frechet density and see if any process suggests itself. If we define the orthogonal function $\theta(\cdot)$ as

$$\theta(z) \text{ is } \left\{ \begin{array}{ll} \text{positive} & |z| \le a \\ \text{negative} & |z| > a \end{array} \right\}$$

then the resulting Frechet density would have the qualitative aspects of Figure 1.



Shaded regions have higher-than-usual probability of occurence.

Figure 1.

It is clear that there is a tendency for X and Y both to be large or both to be small. Let X and Y be adjacent samples in a noise sequence $\{N_t\}$. The behavior given by

$$|N_t| \text{ large} \Longrightarrow |N_{t+1}| \text{ large}$$
$$|N_t| \text{ small} \Longrightarrow |N_{t+1}| \text{ small}$$

is suggestive of a model developed to describe a type of underwater noise [6]. The submarine arctic acoustics are sometimes characterized by high-volume ice-cracks which punctuate low-level background noise. To model this environment Czarnecki [7] and others used two Gaussian sources, one of low power and the other high; the observations process would be formed by switching between these. Such a system is shown in Figure 2, where $f(\cdot)$ and $g(\cdot)$ are respectively high- and low-power Gaussian densities.

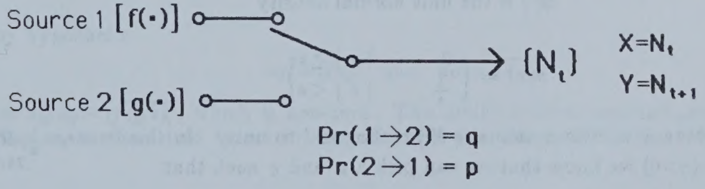


Figure 2.

The switch would be governed by a two-state Markov process. If the switching sequence were known exactly, the bivariate density of adjacent samples would be the product of $\{f(n_t) \text{ or } g(n_t)\}$ with $\{f(n_{t+1}) \text{ or } g(n_{t+1})\}$; however, such a situation is not what we wish to examine. What is assumed in this case is that t is some randomly-chosen time and that we wish to find the density of N_t and N_{t+1}. This will then be some average density over the various modes of operation.

Qualitatively this density will have the attributes shown in Figure 1. If the switch acts sufficiently seldom there will be a tendency for the process to get "stuck" in a mode for a while, suggesting that it would be likely (or at least more likely than if the samples came from an i.i.d. mixture model) that successive samples both be large or both small. In fact, it can be shown that this density is indeed Frechet. Furthermore, and this is the main point of the section, *any* observations process derived from the more general model in Figure 2 can be shown to have Frechet bivariate densities of adjacent samples.

The bivariate density $f(x,y)$ can be written as

$$f(x,y) = h(x)h(y)[1+\gamma\theta(x)\theta(y)]$$

where $h(\cdot)$ is the common univariate density, that is

$$h(\cdot) = \frac{p}{p+q} f(\cdot) + \frac{q}{p+q} g(\cdot)$$

$$\gamma = pq(1-(p+q))$$

$$\theta(\cdot) = \frac{1}{p+q}\left[\frac{f(\cdot)-g(\cdot)}{h(\cdot)}\right]$$

and of course where $f(\cdot)$, $g(\cdot)$, p, and q are determined by the underlying physical model. The details are shown in Appendix C. Furthermore, and this is not shown in the appendix, it can be proven that the bivariate density of N_t and N_{t+k} is *also* Frechet. The form of the representation is the same as in the adjacent- samples case, except now

$$\gamma = pq[1-(p+q)]^k$$

It is interesting to note that since $\theta(\cdot)$ is not normalized, the process is not in general Markov.

The Density and its Underlying Model

It has been shown that for the physical "switched" mechanism of Figure 3 a Frechet bivariate density can be derived. Possibly more interesting is the fact that for every symmetric Frechet density a "switched" model can be found. This result is shown in Appendix D, and is constructive. We can now say that a density is symmetric Frechet if and only if it comes from a switched model. An example follows.

Let us write a known bivariate density

$$f(x,y) = \phi(x)\phi(y)\left[1+\tfrac{1}{2}\theta(x)\theta(y)\right]$$

where

$\phi(\cdot)$ is the unit normal density

and

$$\theta(z) = \begin{Bmatrix} 2 & |z| \geq a \\ -\frac{1}{4} & |z| < a \end{Bmatrix}$$

and of course where a is chosen to make $\theta(\cdot)$ orthogonal to unity. In this case $\gamma=\frac{1}{2}$; following Appendix D ($\gamma>0$) we know that we can pick a p and q such that

$$\left(\frac{1}{\max\theta}\right)\left(\frac{-1}{\min\theta}\right)\left[1-(p+q)\right] = \frac{1}{2}$$

which in this case reduces to

$$p+q = \frac{3}{4}$$

Thus we know that

$$p=\frac{1}{12} \quad q=\frac{2}{3} \quad \Delta=6$$

With this in mind we can construct the underlying densities

$$f(\cdot) = \phi(\cdot)\left[1+4\theta(\cdot)\right]$$

$$g(\cdot) = \phi(\cdot)\left[1-\tfrac{1}{2}\theta(\cdot)\right]$$

Conclusion

In this paper we have explored the applicability of a particularly manageable bivariate density, the Frechet. It was shown that many densities, for example the bivariate normal, are not Frechet. It was also noted that over this class Pearson's contingency is upper-bounded by unity, implying that the Frechet representation cannot describe random variables which are too dependent.

A system was introduced which gave as its output a sequence of Frechet-distributed random variables. This system consisted of two i.i.d. sources and a memoryless switch deciding which of the two had control of the output. It was further shown that such a system underlies every Frechet density.

Appendix A

It will be shown in this appendix that densities with multi-term canonical series expansions cannot be Frechet. Let us write a bivariate density as

$$f(x,y) = g(x)g(y)\left[1 + \sum_{i=1}^{\infty} c_i\,\eta_i(x)\eta_i(y)\right]$$

where

$g(\cdot)$ is the common marginal
$\{\eta_i(\cdot)\}$ are functions orthonormal to $g(\cdot)$

and further require that $c_i \neq 0$ for more than one value of i, call two of these values M and N. Assume that we also can write

$$f(x,y) = g(x)g(y)\left[1+\theta(x)\theta(y)\right]$$

The function $\theta(\cdot)$ is orthogonal to unity, and so can be represented as

$$\theta(z) = \sum_{i=1}^{\infty} a_i\,\eta_i(z)$$

Thus, comparing the two representations, we have that

$$a_i\,a_j = \left\{\begin{array}{cc} c_i & i=j \\ 0 & else \end{array}\right\}$$

But by hypothesis

$$a_M = \sqrt{c_M} \quad \text{and} \quad a_N = \sqrt{c_N}$$

and so $a_M\,a_N = \sqrt{c_M\,c_N}$, which is non-zero. This contradiction demonstrates that if a density has more than two terms in its canonical (i.e. diagonal) series, then it cannot be Frechet.

Appendix B

Find the maximal contingency possible for a density in the Frechet class. We have

$$\phi^2 = \int\int \frac{f(x,y)^2}{g(x)g(y)}\,dxdy - 1$$

$$\phi^2 = \int\int g(x)g(y)\big(1+\theta(x)\theta(y)\big)^2 dxdy - 1$$

$$\phi^2 = \left[\int g(x)\theta(x)^2 dx\right]^2$$

If one maximizes ϕ (since in this case $\phi > 0$) with respect to $\theta^*(\cdot)$ and subject to

$$\int g(x)\theta^*(x)\,dx = 0$$

$$-1/K \leq \theta^*(x) \leq K$$

we see that $\theta^*(x)=K$ or $-1/K \quad \forall x$

Some further calculation shows us that regardless of the specific choice for $\theta^*(\cdot)$

$$\phi^2 = 1$$

It is also true that this result could have been shown through use of the canonical series; such a proof would not have been constructive, however.

Appendix C

In this appendix we will find an expression for the bivariate density of two adjacent variables in the switched-source system defined in the text. Let the system be as shown in Figure 2, and write the transition matrix of the switch as

$$\mathbf{P} = \begin{bmatrix} 1-q & p \\ q & 1-p \end{bmatrix}$$

where $\mathbf{P}_{ij}$ is the probability of switching from source j to source i in successive samples. It can be seen that

$$f(x,y) = \sum_{i=1}^{2} \mathbf{Pr}(Y=y \mid X=x, \text{X from i}) \mathbf{Pr}(X=x \mid \text{X from i}) \mathbf{Pr}(\text{X from i})$$

We now develop expressions for the three quantities on the right side.

For the first term we can write

$$\mathbf{Pr}(Y=y \mid X=x, \text{X from i}) = \mathbf{Pr}(Y=y \mid \text{X from i})$$

and so

$$\mathbf{Pr}(Y=y \mid X=x, \text{X from i}) = \begin{cases} (1-q)f(y)+qg(y) & i=1 \\ pf(y)+(1-p)g(y) & i=2 \end{cases}$$

By definition the second term can be written as

$$\mathbf{Pr}(X=x \mid \text{X from i}) = \begin{cases} f(x) & i=1 \\ g(x) & i=2 \end{cases}$$

The third term can be found by noting that the switch controls a two-state Markov process. The stationary probability densities can be computed as the eigenvectors of the transition matrix corresponding to the unity eigenvalue. In this case

$$\mathbf{Pr}(\text{X from i}) = \begin{cases} \dfrac{p}{p+q} & i=1 \\[2ex] \dfrac{q}{p+q} & i=2 \end{cases}$$

The expression for the bivariate density thus becomes

$$f(x,y) = \frac{p(1-p)}{p+q} f(x)f(y) + \frac{pq}{p+q}\Big[f(x)g(y)+f(y)g(x)\Big] + \frac{q(1-p)}{p+q} g(x)g(y)$$

In order to get the density in standard Frechet form it is necessary at this point to introduce $h(\cdot)$, the univariate density of the process:

$$h(\cdot) = \frac{p}{p+q} f(\cdot) + \frac{q}{p+q} g(\cdot)$$

This expression may be substituted for $f(\cdot)$, and after considerable algebra we can write

$$f(x,y) = \frac{(1-q)(p+q)}{p} h(x)h(y) + \frac{q[1-(p+q)]}{p}\Big[g(x)g(y)-h(x)g(y)-g(x)h(y)\Big]$$

Some further simplification gives the result mentioned in the text, that is

$$f(x,y) = h(x)h(y)\left\{1 + \frac{pq\,[1-(p+q)]}{(p+q)^2}\left[\frac{f(x)-g(x)}{h(x)}\right]\left[\frac{f(y)-g(y)}{h(y)}\right]\right\}$$

It should be noted that

$$\int\left[\frac{f(z)-g(z)}{h(z)}\right]h(z)dz = \int\left[f(z)-g(z)\right]dz$$

which verifies the orthogonality condition on the θ's. The other Frechet condition, that of positivity, must hold due to validity of the bivariate density.

Appendix D

In this appendix it will be shown that any symmetric Frechet density can be said to come from the switched model presented in the text and in Appendix C.

Suppose we know the Frechet density $f_F(x,y)$

$$f_F(x,y) = h(x)h(y)\left[1+\gamma\theta(x)\theta(y)\right]$$

and of course we know $h(\cdot)$, γ, and $\theta(\cdot)$. We know from previous work that if a density $f_{SM}(x,y)$ comes from the switched model shown in the text in Figure 2 we can write

$$f_{SM}(x,y) = h(x)h(y)\left[1+\alpha\psi(x)\psi(y)\right]$$

where

$$h(\cdot) = \frac{p}{p+q}f(\cdot)+\frac{q}{p+q}g(\cdot)$$

$$h(\cdot)\psi(\cdot) = \frac{1}{p+q}\left[f(\cdot)-g(\cdot)\right]$$

$$\alpha = pq\left[1-(p+q)\right]$$

The challenge is to find valid $p,q,f(\cdot)$ and $g(\cdot)$ such that

$$f_{SM}(x,y) = f_F(x,y)$$

The first thing to notice is that $\theta(\cdot)$ and $\psi(\cdot)$ are just multiples of each other. Define Δ such that

$$\Delta = \frac{\psi(\cdot)}{\theta(\cdot)}$$

Secondly, from the previous expressions for $h(\cdot)$ and $h(\cdot)\psi(\cdot)$, we can find $f(\cdot)$ and $g(\cdot)$:

$$f(\cdot) = h(\cdot)[1+q\,\Delta\theta(\cdot)]$$

$$g(\cdot) = h(\cdot)[1-p\,\Delta\theta(\cdot)]$$

These $f(\cdot)$ and $g(\cdot)$ certainly integrate to unity, so to be valid densities all they need is positivity. The problem can be re-stated as follows:

Find p,q, and Δ such that

1 $\quad \gamma = pq\left[1-(p+q)\right]\Delta^2$

2 $\quad 0 \le p,q \le 1$

3 $\quad p\,\Delta \le \dfrac{1}{\max\theta}$ and $q\,\Delta \le \dfrac{-1}{\min\theta}$

It will be helpful to remember that from the positivity of the original density we can write

$$\mathrm{Max}\left\{\frac{-1}{(\max\theta)^2},\frac{-1}{(\min\theta)^2}\right\} \le \gamma \le \left(\frac{1}{\max\theta}\right)\left(\frac{-1}{\min\theta}\right)$$

The remainder of the proof is most easily accomplished in two parts, one for $\gamma<0$ and one for $\gamma \geq 0$.

The Case $\gamma \geq 0$

In this case the number γ is known and bounded by

$$0 \leq \gamma \leq \left(\frac{1}{\max\theta}\right)\left(\frac{-1}{\min\theta}\right)$$

Without loss of generality, let

$$|\max\theta| \leq |\min\theta|$$

Relate p, q, and Δ in the manner

$$p = \frac{1}{\Delta(\max\theta)} \qquad q = \frac{-1}{\Delta(\min\theta)}$$

to satisfy Condition 3 and leave Δ unchosen for now. From this we can re-write Condition 1 as

$$\gamma = \left(\frac{1}{\max\theta}\right)\left(\frac{-1}{\min\theta}\right)\left[1-(p+q)\right]$$

We now have a value for $(p+q)$, which due to the inequality for γ is bounded as

$$0 \leq (p+q) \leq 1$$

It is possible to solve for p, q and Δ and be certain that Condition 2 is satisfied, since both p and q are positive and $p+q \leq 1$

The Case $\gamma<0$

The number γ is known and bounded by

$$\mathrm{Max}\left\{\frac{-1}{(\max\theta)^2}, \frac{-1}{(\min\theta)^2}\right\} \leq \gamma \leq 0$$

Without loss of generality, let $|\max\theta| \geq |\min\theta|$, so that we can write

$$\frac{-1}{(\max\theta)^2} \leq \gamma \leq 0$$

Thus choose

$$\Delta = \frac{1}{\max\theta}$$

This means that since $0 \leq p, q \leq 1$ we can write

$$p\,\Delta \leq \frac{1}{\max\theta}$$

$$q\,\Delta \leq \frac{1}{\max\theta} \leq \frac{-1}{\min\theta}$$

and so Condition 3 is satisfied.

Now with this choice of Δ it is true that

$$-1 \leq \frac{\gamma}{\Delta^2} \leq 0$$

Thus, since for $0 \leq p, q \leq 1$ the function $pq\,[1-(p+q)]$ takes on all values between -1 and $\frac{1}{3}$, it is possible to choose p and q such that Conditions 1 and 2 are satisfied.

It should be stressed that the above proof is only of existence; it does not seem to be true that there is a *unique* physical model underlying every symmetric Frechet density.

Acknowledgment

This work was supported in part by the U.S. Army under Grant DAAG29-85-K-0191 and in part by the National Science Foundation under Grant ECS83-17777.

REFERENCES

[1] H. O. Lancaster, *The Chi-Square Distribution* ,Wiley, 1969.

[2] H. O. Lancaster, "The Structure of Bivariate Distributions", *Annals of Mathematical Statistics* , Vol 29, pp. 719-736, 1958.

[3] M. Frechet, "Sur les Tableau de Correlation dont les Marges sont Donnees", *Ann. Univ. de Lyon* , ser. 3, A14, pp. 53-77, 1951.

[4] G. Moustakides, *The Representation of Bivariate Densities with Applications in Detection and Estimation.* , PhD Thesis Princeton University, 1983.

[5] G. Moustakides and J.B. Thomas, "Min-Max Detection of Weak Signals in ϕ-Mixing Noise", *IEEE Transactions on Information Theory* , Vol IT-30, No. 3, pp. 529-537, 1984.

[6] R. Dwyer, "FRAM II Single Channel Ambient Noise Statistics", *Paper 101st Meeting of Acoustical Society of America",* 1981.

[7] S. Czarnecki, *Nearly Optimal Detection of Signals in Non-Gaussian Noise* , PhD Thesis Princeton University, 1983.

INCLUSION RELATIONSHIPS AMONG PERMUTATION PROBLEMS

Daniel P. Bovet and Alessandro Panconesi
Dipartimento di Matematica, Università
di Roma "La Sapienza", Piazzale Aldo
Moro 5, 00185 Rome, Italy

ABSTRACT: Let $X = \{x_1,\ldots,x_n\}$ be a set of n elements. A permutation problem of X is a decision problem in NP which consists of determining whether there exists an n element permutation satisfying a set of ordering constraints expressed as collections of permutations of at most k elements of X. Several well known NP-complete problems such as cyclic ordering, betweeness, and serializability can be expressed as permutation problems.

If we consider symbol preserving polynomial reductions, i.e., reductions that do not make use of additional elements, strict inclusion relationships among permutation problems can be proved. In particular, the serializability problem is shown to be included in the deadlock avoidance one.

Keywords: NP-complete problems, inclusion relationships, serializability, deadlock avoidance.

1. INTRODUCTION

Two well known combinatorial problems, denoted in the literature as *serializability* (of database histories) and *deadlock avoidance* are apparently quite different, although they both refer to a set of active elements (transactions or processes) competing for a common pool passive ones (entities or resources).

The serializability problem can be formulated as follows a set $T_1,\ldots,T_n$ of transactions access concurrently a database consisting of entities $e_1,\ldots,e_m$. The *state* of a database is the set of values of the m entities at a given time. Each transaction T_i consists of a sequence of elementary Read or Write operations on sets of entities. An *history* h relative to transactions $T_1,\ldots,T_n$ is any permutation of all Read and Write operations of the n transactions which preserves the internal ordering of elementary operations within each transaction. Two histories h and h' applied to the database in state S are said to be *equivalent* if they both transform it into the same state S'. An history is *serial* if it corresponds to the serial execution of transactions $T_{i1},T_{i2},\ldots,T_{in}$. The *serializability problem* (in short, SR) consists of deciding whether an history h admits an equivalent serial history h_s.

The deadlock avoidance problem, in turn, is defined as follows: denote with $P_1,\ldots,P_n$ a set of *processes* and with $R_1,\ldots,R_m$ a set of *resources*. Each R_j includes a number $k_j \geq 0$ of *units*. For each process P_i there are two m-component vectors denoted as $REQ_i(r_{i1},\ldots,r_{im})$ and $PROD_i(p_{i1},\ldots,p_{im})$ with $r_{ij} \geq 0$ and $p_{ij} \geq 0$, for $1 \leq j \leq m$.

Each P_i requests first r_{ij} units of each R_j and, after obtaining all r_{ij} units in $REQ_i(r_{i1},\ldots,r_{im})$, produces p_{ij} units for each R_j.

The *deadlock avoidance problem* (in short, DA) consists of deciding whether there exists a sequence of resource units assignments to processes satisfying all their requests (this problem is also referred to as finding a deadlock-free execution sequence for the n processes).

Both problems are known to be NP-complete [6,3], although it is not clear whether one of them is inherently more complex than the other, that is, whether there exists an inclusion relationship under some acceptable complexity measure.

It can be observed that the main difference between SR and DA refers to resource attributes: in SR, each resource is available in a single unit and is modifiable in the sense that successive Write operations on the same resource (entity) alter its value. In DA, resource units are not modifiable; however, the number of units of any resource may vary, depending on the requests and productions of processes acting upon it.

From a different perspective, it may also be said that each Write operation on an entity corresponds to the destruction of its current value and the creation of a new value. Similarly, each destruction or creation of resource units in DA can be viewed as the modification of an entity k_i denoting the number of available units in R_i.

In this paper, it is shown that SR is included in DA: in order to support this claim, an inclusion relationship between NP problems must be properly defined. This is done in section 2 where the two problems considered are restated as "permutation problems", that is, decision problems referring to the existence of a permutation of n elements, subject to a set of ordering constraints. This approach turns out to be fruitful: in section 3 inclusion relationships among DA, SR and a few other apparently different. NP-complete problems are proved. Finally, in section 4, some observations are made on the results of the previous sections.

2. PERMUTATION PROBLEMS

Let $X = \{x_1,\dots,x_n\}$ be a set of n elements; denote with $C_i \subset X$ a subset of X and with $\Pi^*(C_i)$ the set of all possible permutations of elements of C_i. Let $\Pi^\circ(C_i)$ a subset of $\Pi^*(C_i)$. A permutation π of elements of X *agrees with* $\Pi^\circ(C_i)$ if, after removing from π all elements not included in C_i, the remaining permutation belongs to $\Pi^\circ(C_i)$.

Definition 1. A *permutation problem* is a decision problem in NP having the following properties:

1) It includes a set X of n elements and a collection $C = \{C_1,\dots,C_m\}$ of subsets of X;
2) The cardinality $|C_i|$ of any C_i is smaller than or equal to a constant k;
3) It includes a set of permutations $\Pi^\circ(C_i)$ for every C_i;
4) The question to be answered is: does there exist a permutation π of elements of X which agrees with all $\Pi^\circ(C_i)$?

Intuitively, algorithms for solving permutation problems must decide whether there exists an ordering among n elements satisfying a set of ordering constraints; an example of permutation problem is CYCLIC ORDERING [1]: given a finite set X and a collection C of triples $C_i = (x_j, x_k, x_l)$ of distinct elements of X, is there a permutation π of elements of X such that, for each C_i, elements x_j,x_k,x_l appear in it the order x_j,x_k,x_l or x_k,x_l,x_j or $x_l x_j x_k$?

This problem can be easily reformulated as a permutation problem by introducing $\Pi^\circ(C_i) = \{x_j x_k x_l,\ x_k x_l x_j,\ x_l x_j x_k\}$ and by replacing the previous question with: does there exist a permutation π agreeing with all $\Pi^\circ(C_i)$?

Definition 2. Let P and P' be two permutation problems in NP. A polynomial reduction f between P and P' is said to be a *symbol preserving reduction* in short SPR, if the instances of P' obtained from P according to f have exactly the same symbols.

If f is a SPR between P and P', any n-symbol instance of P is mapped into an equivalent n-symbol instance of P'; the converse is not true in general (see section 3). Informally speaking, the "descriptive power" of P' is at least as great as the one of P. We can then introduce the following:

Definition 3. Let P and P' be two permutation problems. P is *included* in P', in symbols $P \subset P'$, if there exists a SPR f from P to P'.

In the next section, we shall consider permutation problems whose sets C_i include three elements and whose corresponding $\Pi^\circ(C_i)$ represent some kind of ordering constraints.

Six types of $\Pi^\circ(C_i)$ will be considered; the corresponding ordering constraints will be called: *symmetric* and *asymmetric betweenness*, *symmetric* and *asymmetric nonbetweenness*, *cyclic ordering*, and *serializability*. They will be denoted, respectively, as: B(a,bc),b(a,bc),NB(a,bc),nb(a,bc),CO(abc), and SR(a,bc). Each ordering constraint on three elements a,b,c yields a set $\Pi^\circ(\{a,b,c\})$ as defined in fig. 1. Each column corresponds to an ordering constraints and each row to a permutation; a permutation in row i is included in set $\Pi^\circ(\{a,b,c\})$ of the ordering constraint of column j if entry (i,j) has value 1.

3. INCLUSION RELATIONSHIPS AMONG PERMUTATION PROBLEMS

The six ordering constraints of fig. 1 give raise to six corresponding NP-complete permutation problems: three of them, namely, Betweenness (B), Cyclic Ordering (CO), and Serializability (SR) are already known [5,1,6]; the remaining three which we call Symmetric Non Betweenness (NB), Asymmetric Betweenness (b), and Asymmetric Non Betweenness (nb) are new.

It is easy to verify that B and CO are permutation problems: as an example, Betweenness can be reformulated as follows:

Instance: a set $X = \{x_1,\dots,x_n\}$ of elements and a collection $\{B_j(a,bc)\}$ of symmetric betweenness constraints.

Question: is there a permutation π of the n x_i agreeing with all B_j?

Similarly, CO,NB,b,nb can be expressed as permutation problems using the corresponding ordering constraints of fig. 1. Let us now turn our attention to SR.

Serializability: As it has been shown in the literature, given a history h formed by n transactions T_i, it is possible to derive from it a set of constraints of two types: the first are ordering relationships between pairs of transactions of the form "T_i must precede T_j" (in symbols, $T_i < T_j$) in every serial history equivalent to h. The second are ordering relationships between triples of transactions of the form: "T_i must precede T_j" *and* "T_k must either precede T_i or follow T_j" in every serial history equivalent to h. This last relationship corresponds to:

$$(T_i < T_j) \wedge \{(T_k < T_i) \vee (T_j < T_k)\}$$

which is equivalent to $SR(T_k,T_iT_j)$ defined in fig. 1.

SR is solved [6] by "framing" history h with two additional transactions T_0 and T_f: the first must precede all remaining ones while the second must follow them ($T_0 < T_i, T_i < T_f$). Using T_0 and T_f, it is possible to express relationships of the form $T_i < T_j$ by means of SR(a,bc) relationships; this is done as follows: $T_0 < T_i$ is replaced with $SR(T_f,T_0T_i)$, $T_i < T_j$ by $SR(T_0,T_iT_j)$, and $T_i < T_f$ by $SR(T_0,T_iT_f)$.

Thus, SR reduces to the following permutation problem:

Instance: a set $X = \{T_1,\ldots,T_n\}$ of n elements and a collection $\{SR_h(T_k,T_iT_j)\}$ of serializability constraints.

Question: is there a permutation of the n T_i elements agreeing with all SR_h constraints?

Let us now consider inclusion relationships (see section 2) among the six problems considered.

Lemma 1. B is included both in NB and in b.

Proof. In order to prove: $B \subset b$, we observe from fig. 1 that each constraint B(a,bc) of B is equivalent to the pair of constraints b(a,bc) and b(a,cb); in symbols: $B(a,bc) = b(a,bc) \wedge b(a,cb)$.

Similarly, the equivalence $B(a,bc) = NB(b,ac) \wedge NB(c,ab)$ proves that $B \subset NB$. ☐

Lemma 2. CO is included in b.

Proof. $CO(a,b,c) = b(a,cb) \wedge b(b,ac)$. ☐

Lemma 3. SR,NB, and b are included in nb.

Proof. We observe from fig. 1 that nb forbits permutation bac. Permuting the three letters a,b,c inside nb(x,yz), we can forbid any of the six permutations. Since all remaining ordering constraints forbid two or more permutations, they are equivalent to a conjunction of nb constraints. ☐

Lemma 4. Inclusions of Lemmas 1,2 and 3 are strict.

Proof. Let us consider an elementary instance of P' (P' = NB,b in Lemma 1; P' = b in Lemma 2; P' = nb in Lemma 3) consisting of three symbols together with one ordering constraint: such instance cannot be represented in P (P = B in Lemma 1, P = CO in Lemma 2; P = SR,NB,b in Lemma 3) without adding extra symbols. ☐

Lemma 5. The only inclusion relationships between the six permutation problems considered are those illustrated in lemmas 1,2,3.

Proof. By exhaustive search. In order to prove that problem P is not included in problem P', it is sufficient to verify that an ordering constraint

$A(x_1x_2x_3)$ of P cannot be obtained as a conjunction of any subset $\{B(x_{i1}x_{i2}x_{i3})\}$ of constraints of P'. □

Before turning our attention on the relationship between SR and DA, we investigate whether DA can be reformulated as a permutation problem.

Deadlock avoidance: As shown in [4], it is easy to verify that every solution for DA can be expressed as a permutation $P_{i1},\ldots,P_{in}$ of processes (execution sequences) assuming that, first, only requests of P_{i1} are satisfied, then only those of P_{i2}, and so on.

For sake of simplicity, we consider an NP-complete subcase of DA, which will be denoted as DA*, such that:

1) all request and production vectors REQ_i and $PROD_i$ are binary vectors;

2) all k_i units of R_i are initially set to 0 or 1;

3) each resource is requested at most by two processes;

4) each resource is produced at most by two processes.

Lemma 6. DA* is a permutation problem.

Proof. Consider first a simple instance of DA* with two resources R_1 and R_2 and with $k_1 = k_2 = 0$. The n processes competing for the two resources are classified as consumers, producers or keepers of resource R_i. A process is called a *consumer* of R_i ($i = 1,2$) if it requests one unit of R_i and produces 0 units of R_i; a *producer* if it requests 0 units of R_i and produces one unit of R_i; a *keeper* if it requests and produces 1 unit of R_i.

Denote with $Q_1,\ldots,Q_{s1}$ ($Q'_1,\ldots,Q'_{s2}$) the s_1 (s_2) consumers of R_1(R_2), with $P_1,\ldots,P_{t1}$ ($P'_1,\ldots,P'_{t2}$) the t_1 (t_2) producers, and with $K_1,\ldots,K_{u1}$ ($K'_1,\ldots,K'_{u2}$) the u_1(u_2) keepers.

The n processes considered play the role of set X in point 1 of definition 1. The collection C consists of:

$$C_1 = \{Q_1,\ldots,Q_{s1},P_1,\ldots,P_{t1},K_1,\ldots,K_{u1}\} \text{ and } C_2 = \{Q'_1,\ldots,Q'_{s2},P'_1,\ldots,P'_{t2}K'_1,.\,K'_{u2}\}$$

Clearly, conditions 3 and 4 in the definition of DA* insure that C_1 and C_2 satisfy point 2. The two sets $\Pi^\circ(C_1)$ and $\Pi^\circ(C_2)$ consists of all permutations of the $s_i+t_i+u_i$ elements of C_i ($i = 1,2$) satisfying the following condition:

"If a consumer or a keeper of R_i appears as the h-th element of a permutation in $\Pi^\circ(C_i)$, then the first h-1 elements must include more producers than consumer of R_i".

A solution S of the instance must be an n process permutation agreeing with all $\Pi^\circ(C_i)$ because a process in S can occupy the h-th position only if the resources it requests have been produced previously; however, this is the only condition that must be satisfied by solutions of DA* problems, thus any S agreeing with all $\Pi^\circ(C_i)$ is a solution. This establishes points 3 and 4.

Finally, the problem formulation given for DA^* associates a distinct $\Pi^{\alpha}(C_i)$ with every resource R_i; thus, the above result can readily be extended to m-resource systems. □

DA cannot be formulated as a permutation problem (see Def. 1): if conditions 3) and 4) in the definition of DA^* do not hold, using the same construction of Lemma 6, we associate to each R_i the set of processes C_i which are consumers, producers or keepers of R_i. However, because 3) and 4) have been dropped, the cardinality of a set C_i may vary arbitrarily between 1 and n. This establishes the following:

Lemma 7. DA is not a permutation problem.

The relationship between DA^* and the six permutation problems previously considered is stated in the next lemma and theorem.

Lemma 8. nb is included in DA^*.

Proof. any instance of nb can be translated into an instance of DA^* in the following simple way: replace any ordering constraint nb(a,bc) with a resource $R_{a,bc}$ with 1 unit available and replace symbols a,b,c with the corresponding processes A,B,C. Using the definition of Lemma 6, process B is a consumer of $R_{a,bc}$, C is a producer of $R_{a,bc}$, and A is a keeper for the same resource (see fig. 2). As a simple check reveals, the set of permutation allowed by the construction of fig. 2 coincides with the set of permutations agreeing with nb(a,bc). □

The results obtained so far can be summarized in:

Theorem 1. The inclusion relationships among the six permutation problems considered and DA^* are represented in fig. 3.

4. FINAL REMARKS

Let us make some remarks with respect to fig. 2. The number of 1's in a column characterizing a 3-element relation in fig. 1 is, in some sense, a measure of the complexity of the corresponding problem. This appears in the proofs of lemmas 1-4.

In general, a 3-element ordering constraint can be defined by means of a truth table as the one shown in fig. 1. Truth tables including a single 1 give raise to a well known polynomial problem, the acyclicity of a directed graph. It turns out that the only NP-complete problems with two 1's are B and SR, the remaining ones are polynomial. Polynomial problems are still present in truth tables with 3 1's while, with 5 1's, the only possible problem in nb.

It is also interesting to observe that considering serializability as a permutation problem allows us to produce a rather simple NP-completeness proof of SR using the reduction technique presented in [3].

In that paper, the satisfiability problem (an exhaustive discussion of satisfiability can be found in [2]) is reduced polynomially into a DA problem with two types of resources corresponding, respectively, to boolean variables and clauses. Introducing additional resources and processes, the DA problem can be easily reduced into a DA^* problem such that DA is deadlock-free iff the corresponding DA^* is deadlock-free.

In the resulting DA^*, all process interactions on boolean variable resources are simple interactions shown, respectively, in fig. 4A and 4B. Thus, deriving from DA^* an history h^* such that DA^* is deadlock-free iff h^*

is serializable can be done quite easily.

REFERENCES

1. Z. Galil and N. Megiddo, *Cyclic ordering is NP-complete,* Theor. Comput. Sci., 5 (1977), pp. 179-182.
2. M.R. Garey and D.S. Johnson, *Computers and Intractability. A guide to the Theory of NP-Completeness,* W.H. Freeman, San Francisco, 1979.
3. E.M. Gold, *Deadlock protection: easy and difficult cases,* SIAM J. Comput., 7 (1978), pp. 320-336.
4. R.C. Holt, *Some deadlock properties of computer systems,* ACM Comput. Surveys, 4 (1972), pp. 179-196.
5. J. Opatrny, *Total ordering problem,* unpublished manuscript, (1978).
6. C.H. Papadimitriou, *The serializability of concurrent database updates,* J. ACM, 26 (1979), pp. 631-653.

	B(a,bc)	b(a,bc)	NB(a,bc)	nb(a,bc)	CD(abc)	SR(a,bc)
abc	0	0	1	1	1	1
acb	0	1	1	1	0	0
bac	1	1	0	0	0	0
bca	0	0	1	1	1	1
cab	1	1	0	1	1	0
cba	0	1	1	1	0	0

Fig. 1. Definitions of the six ordering constraints.

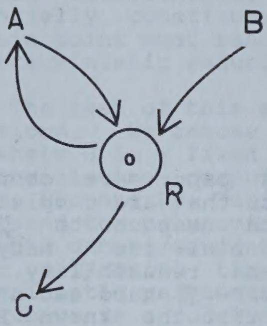


Fig. 2. Transformation of nb into DA*.

Permutation BAC is forbidden, the remaining permutations are allowed. (Requests are denoted as directed arcs (P,R) and productions as directed arcs (R,P))

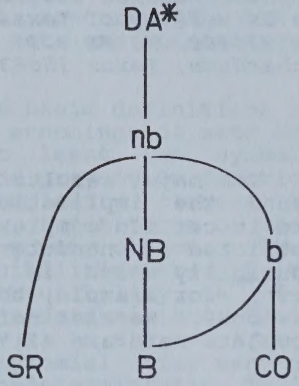
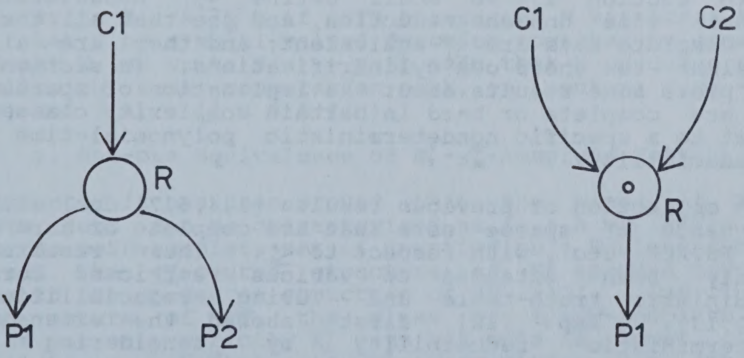


Fig. 3. Inclusion relationships among permutation problems.



Interactions on boolean variable resources

Interactions on clause resources

Fig. 4. Process interactions on a simplified DA* problem.

NONDETERMINISTIC REDUCIBILITIES AND SPARSE COMPLETE SETS

HSI-LIN SUNG
Computer Science Program
The University of Texas at Dallas
Box 830688, MS FN 3.3
Richardson, Texas 75083-0688

Abstract

The major results we prove in this paper are centered around the implications of sparse sets that are complete or hard in certain complexity classes with respect to $\leq_{dm}^{NP}$, a restricted nondeterministic polynomial-time many-one reducibiliy which is weaker than gamma reducibility. We prove, for example, that there is no coNP-$\leq_{dm}^{NP}$-hard set unless NP = coNP. We also define $\leq_{1}^{NP}$, and show that the known Σ_2^P-$\leq_m^{NP}$-complete sets are all $\leq_1^{NP}$-equivalent.

1. Introduction

A set is (polynomially) sparse if there exists a polynomial p such that for all n the number of strings of length up to n in the set is bounded by p(n). The study of sparse sets was originally motivated by the famous Hartmanis-Berman Conjecture that all NP-complete sets are polynomially isomorphic [2]. Since all the known NP-complete problems have polynomial time padding functions that allow an arbitrary instance of the problem to be padded into exponentially many different variants of the same problem whose lengths are all polynomially related to the original, the Hartmanis-Berman Conjecture implies that no sparse sets can be NP-complete. Another direct implication of the conjecture is that P ≠ NP.

In section 2, we shall define $\leq_1^{NP}$, nondeterministic polynomial-time one-one reduction, and see that all the known Σ_2^P-$\leq_m^{NP}$-complete sets are $\leq_1^{NP}$-equivalent, and they are all $\leq_1^{NP}$-equivalent to their own cylindrifications. In section 3, we shall prove some results about the implication of sparse sets that are complete or hard in certain complexity classes with respect to a specific nondeterministic polynomial-time many-one reducibility.

A collection of previous results [1,4,6,7] deal with the consequence of sparse sets that are complete or hard in NP, coNP, PSPACE, etc., with respect to $\leq_m^P$. These results have recently been extended to various restricted forms of deterministic truth-table and Turing reducibilities in [11,12,13]. Yap [12] first showed the extension to nondeterministic reducibility by considering gamma reducibility. The reducibility we consider in section 3 is a restricted form of $\leq_m^{NP}$, which is more general than gamma reducibility. The proofs of our new results adapt the tree search technique pioneered by P. Berman and Fortune [1,4] to accommodate the situation where a nondeterministic reduction, instead of a deterministic one, maps a set which is complete or hard in a certain class, to a sparse set.

We believe that our results, besides showing the implications of sparse sets that are complete or hard, with respect to $\leq_{m}^{NP}$, in various complexity classes, also indicate that researching the significance of sparse sets under nondeterministic reducibilities is an interesting issue which deserves more investigation. This is so because that the various deterministic and nondeterministic reducibilities essentially constitute a two-dimensional spectrum [5], but up to this point most researches on sparse sets are concentrated on deterministic reducibilities.

The rest of this section gives some basic definitions and notations. We assume familiarity with encodings of sets over G*, where G is a fixed alphabet with at least two symbols; polynomial-time computable pairing and projection functions <-,->, P1, and P2; Turing machines; complexity classes NP, coNP, and the polynomial-time hierarchy. Definitions of these concepts can be found in [2,3,5,9], among many others. Three major polynomial time many-one reducibilities are: $\leq_m^P$, the deterministic polynomial time many-one reducibility, $A \leq_m^P B$ iff there is a function f, computed by a deterministic Turing machine in polynomial time, such that $(\forall x)\ [x \in A \iff f(x) \in B]$. $\leq_m^{NP}$ is the nondeterministic polynomial time many-one reducibility, $A \leq_m^{NP} B$ iff there is a nondeterministic Turing machine (NDTM) transducer M that runs in polynomial time such that $(\forall x)\ [x \in A \iff (\exists y \in B)$ [M, following some computation path, produces y as output on input x]]. Note here that the definition of $\leq_m^{NP}$ allows the NDTM transducer to produce an output $y \notin B$ for an input $x \in A$; and it is possible that all computation paths of the transducer either do not halt or halt without outputs, when given $x \in \overline{A}$ as input. $\leq_m^{SN}$, usually referred to as gamma reducibility, is the strong nondeterministic polynomial time many-one reducibility, $A \leq_m^{SN} B$ iff there is an NDTM transducer M that runs in polynomial time such that (1) M produces at least one output on every input, and (2) $(\forall x)(\forall y)$ [M produces output y on input $x \implies [x \in A \iff y \in B]]$. For a comparison of these reducibilities, see [5].

We write $M : A \leq_m^{NP} B$ if NDTM transducer M witnesses that $A \leq_m^{NP} B$. Let f be a multi-valued function computed by some NDTM transducer M, we write $M(x) = y$ to mean that M produces output y, following some computation path, on input x; $f(x) = \{y : M(x)=y\}$ and $f(A) = \bigcup\, x \in A\ f(x)$.

2. One-One Equivalence of Σ_2^P-$\leq_m^{NP}$-complete Sets

Recently it has been proved that the class of Σ_2^P-$\leq_m^{NP}$-complete sets can be characterized as $\{A : A \equiv_m^{NP} \overline{SAT}\}$, where $\overline{SAT}$ is the coNP-complete set of unsatisfiable Boolean formulas [3], and the $\leq_m^{NP}$-degree structure of Σ_2^P appears very much similar to the $\leq_m^P$-degree structure of NP [10]. Due to the relaxed nature of $\leq_m^{NP}$, the class of Σ_2^P-$\leq_m^{NP}$-complete sets contains not only strictly Σ_2^P sets but sets in lower classes as well, e.g., $\overline{SAT}$ is also Σ_2^P-$\leq_m^{NP}$-complete. Many other D^P sets ($D^P = \{L1 \cap L2 : L1 \in NP,\ L2 \in coNP\}$ [8]) which have resisted attempts of D^P-$\leq_m^P$-completeness proofs were found to be D^P-$\leq_m^{NP}$-complete. Since $D^P \subseteq \Sigma_2^P$, all D^P-$\leq_m^{NP}$-complete sets are also Σ_2^P-$\leq_m^{NP}$-complete [10].

In light of the analogy that in many ways $\leq_m^{NP}$ seems to play the same role in Σ_2^P that $\leq_m^P$ plays in NP, it is interesting to investigate if the known Σ_2^P-$\leq_m^{NP}$-complete sets are also isomorphic under certain nondeterministic polynomial time computable functions. However, nondeterministically computed functions, by the very nature of nondeterminism, are generally multi-valued, but isomorphism intuitively implies a one-one mapping. Therefore, before we proceed any further a definition is needed for one-one mapping under nondeterministic polynomial time reductions. Suppose $M : A \le_m^{NP} B$, we write $A \le_1^{NP} B$, read A one-one reduces to B in NP-time, if $(\forall\ x \in A)(\forall\ y \overline{\in}\ A)(\exists\ z \in B)\ [x \neq y \Longrightarrow z \in f(x) - f(y)]$, where f is the function defined by M. This is a natural, reasonable way to define nondeterministic one-one reducibility. One reason that this definition is appropriate is that it renders $\le_1^{NP}$ the nice properties of being reflexive, transitive, and cardinality preserving, in the sense that if $A \le_1^{NP} B$ then $|A| \le |B|$. (Reflexivity and cardinality preservation are obvious. To prove transitivity, let $M1 : A \le_1^{NP} B$ and $M2 : B \le_1^{NP} C$, it is easily seen that $M1 \circ M2 : A \le_1^{NP} C$.) Another reason is that it takes into account the fact that when $f : A \le_m^{NP} B$, it is allowable that $f(\bar{A}) = \emptyset$. Therefore it would be quite restrictive and unnatural if the definition also includes the condition $(\forall\ x \in \bar{A})(\forall\ y \in \bar{A})\ (\exists\ z \in \bar{B})\ [x \neq y \Longrightarrow z \in f(x) - f(y)]$. The following two theorems demonstrate that $\le_1^{NP}$ is indeed the nondeterministic analog of $\le_1^{P}$.

Theorem 1a: $A \times G^* \le_1^{NP} A$ (and thus $A \times G^* \equiv_1^{NP} A$, where $A \times G^*$ is called the cylindrification of A) if and only if there exists an NDTM transducer M which satisfies the following two conditions: (i) $(\forall x, y)\ [x \in A \Longleftrightarrow (\exists z \in A)\ [M(x,y) = z]]$, and (ii) $(\forall x1, x2 \in A)\ (\forall y1, y2)\ (\exists z1, z2 \in A)\ [M(x1, y1) = z1$ and $M(x2, y2) = z2 \Longrightarrow [y1 \neq y2 \Longrightarrow z1 \neq z2]]$.

Proof: (if) It is easy to see that $\tilde{M} : A \times G^* \le_1^{NP} A$, where $\hat{M}(x) = M(P1(x),x)$.
(only if) Suppose $\tilde{M} : A \times G^* \le_1^{NP} A$, then $M(x,y) = \tilde{M}(<x,y>)$ is the desired NDTM transducer.

The two conditions of Theorem 1a basically mean that M(-,-) computes a one-one padding function for A. If we strengthen these conditions to require that not only M(-,-) be a one-one padding, but the padded string y can be decoded from M(x,y) as well, then we have:

Theorem 1b: Let A be a set which satisfies the following two conditions: (i) there exists a (multi-valued partial) function p(-,-), computed by some NDTM transducer M, such that $(\forall x, y)\ [x \in A \Longleftrightarrow (\exists z \in A)\ [\hat{M}(x,y) = z]]$, (ii) there exists a (single-valued partial) function d(-), such that $(\forall x, y)\ [d(\hat{M}(x,y)) =$ if $\hat{M}(x,y) = z$ then $y]$. Then if $M : C \le_m^{NP} A$, the NDTM transducer $\tilde{M}$ defined by $\tilde{M}(x) =$ if $M(x)=y$ then $\hat{M}(y,x)$ is such that $\tilde{M} : C \le_1^{NP} A$.

Proof: For every $x \in C$, there exists a $y \in A$ such that $M(x)=y$, implying the existence of z such that $\hat{M}(y,x)=z \in A$. Conversely, suppose there is a y such that $\hat{M}(y,x) \in A$ and $M(x)=y$, then it must be $x \in C$. This verifies that $\tilde{M} : C \le_m^{NP} A$. Now to prove that $\tilde{M}$ is one-one, suppose it is not, then $(\exists x \in C)(\exists y \in C)(\forall z \in A)\ [x \neq y$ and $z \in \overline{g(x)} \cup g(y)]$, where g is the function computed by $\tilde{M}$. For such a pair $x,y \in C$, choose any $z \in g(x) \cap A$, then $z \in g(y)$, and exist $u,v \in A$ such that $u = M(x)$, $v = M(y)$, and $z \in p(u,x) \cap p(v,y)$. Since d(-) is single-valued, $d(z) = d(\hat{M}(M(x),x)) = d(\hat{M}(M(y),y)) = x = y$, contradicting the assumption that $x \neq y$. Therefore, it must be that $\tilde{M} : C \leq_1^{NP} A$.

It is not hard to verify that all the Σ_2^P-$\leq_m^{NP}$-complete sets, including B_2[9], $\overline{SAT}$, and the D^P-$\leq_m^{NP}$-complete sets in [10] do satisfy the two conditions of Theorem 1b, so they all have the same $\leq_1^{NP}$-degree. If all Σ_2^P-$\leq_m^{NP}$-complete sets had the same $\leq_1^{NP}$-degree, then NP $\neq$ coNP, because NP = coNP implies NP = coNP = Σ_2^P, thus the class of Σ_2^P-$\leq_m^{NP}$-complete sets would include both finite and infinite sets in NP, which can not have the same $\leq_1^{NP}$-degree as $\leq_1^{NP}$ is cardinality preserving.

Note that in Theorem 1b above the decoding function d(-) need not be computable in P-time or NP-time, as we did not consider the invertibility of the function defined by M.

3. $\leq_{dm}^{NP}$ and Sparse Complete Sets

Even though we do not conjecture, despite easily verified supporting facts, that all Σ_2^P-$\leq_m^{NP}$-complete sets form a single $\leq_1^{NP}$-degree, it is, nevertheless, instructive to ponder the significance of a sparse set that is $\leq_m^{NP}$-complete in, say coNP or Σ_2^P. In the following, we prove some results of this nature. Before presenting these results, we need to define still another reducibility: Suppose $M : A \leq_m^{NP} B$, we write $A \leq_{dm}^{NP} B$, read A disjointly many-one reduces to B in NP-time, if $f(\overline{A}) \cap f(\overline{A}) = \emptyset$, where f is the function M computes. Obviously, $A \leq_m^{SN} B \Longrightarrow A \leq_{dm}^{NP} B \Longrightarrow A \leq_m^{NP} B$ for all A and B. Our first sparseness result deals with the consequence of a sparse coNP-$\leq_{dm}^{NP}$-hard set and strengthens a result by Yap [12], which is the same theorem except that $\leq_{dm}^{NP}$ is replaced by $\leq_m^{SN}$.

Theorem 2: If coNP is $\leq_{dm}^{NP}$-reducible to a sparse set, then NP = coNP.

Proof: Suppose coNP is $\leq_{dm}^{NP}$-reducible to a sparse set S. Let $M : \overline{SAT} \leq_{dm}^{NP} S$, where S has density bound p, a polynomial. Assume without loss of generality that M always halts in polynomial time, with or without an output. Let q be a polynomial bounding the increases in length under the reduction performed by M. To decide if an input Boolean formula F with n variables is satisfiable, perform the tree search as described in [2,4,6]. The outline of the search procedure is as follows: Place F at the root, each node G in the tree has two children G0 and G1, which are obtained by substituting 0 and 1, respectively, for the leftmost variable of G. The leaves of the tree are constant formulas which are identically true (1) or identically false (0). Beginning at F, successive nodes to be visited are chosen by the depth-first rule. Use the sparse set S to prune the search, i.e., the nodes below a dead node will not be searched, where a dead node is a node with a label in S. The label on each node G is computed nondeterministically by running M on G. If M halts without output, then terminate the search and accept F, i.e., decide that F $\in$ SAT. If M produces any output, this output becomes the label of G and search is continued. Also, the search will be terminated when (1) more than $n + n * p(q(|F|))$ interior nodes (see [6] for the derivation of this number) of the tree have been visited, this can be determined by using a counter, (2) a node in SAT turns up in the search tree, or (3) the label of F is found to be in S, where the rules to infer if a label is in S are: (i) If M(0) = a, then a $\in$ S, where 0 is the identically false formula; (ii) If M(G0) = a0, M(G1) = a1, M(G) = a, and both a0, a1 $\in S$, then a $\in S$.

If case (1) or (2) causes the termination of the search, then decide F $\in$ SAT; if case (3) does, then decide F $\in \overline{\text{SAT}}$. Thus F $\in$ SAT iff every computation or search decides so, and F $\in \overline{\text{SAT}}$ iff at least one search decides so, implying that SAT $\in$ coNP and consequently NP = coNP.

It can be seen now the reason why we use $\leq^{NP}_{dm}$ instead of $\leq^{NP}_{m}$ is the technical difficulty that there appears no obvious way to adapt the Berman-Fortune tree search technique, as done in the above proof, if $f(A) \cap f(\overline{A}) \neq \emptyset$. Because in this case a node, say y, with a descendant leaf in SAT might be pruned prematurely if M, on input y, produces an output, say 'a', in $\overline{S}$, while earlier M has produced the same output on a previously searched node, say x, which is in $\overline{\text{SAT}}$ and 'a' has been inadvertently put into the list of sparse set of labels. This is possible since x was inferred to be in $\overline{\text{SAT}}$, and thus 'a' put into the list of sparse set of labels when the two sons of x were found to be in $\overline{\text{SAT}}$. As a result of this premature pruning of y, the tree search procedure may end up deciding the input formula is in $\overline{\text{SAT}}$ while it is in fact satisfiable. Therefore the conclusion NP = coNP can not be drawn, because NP = coNP implies SAT $\in$ NP $\cap$ coNP, and thus either of the following two conditions must hold: (1) Whenever the input Boolean formula F is satisfiable, the nondeterministic search procedure, taken as an NP-algorithm for $\overline{\text{SAT}}$, must end up deciding F $\in$ SAT; and whenever F $\in \overline{\text{SAT}}$ there must exist at least one search that verifies that. (2) The search procedure is taken as an NP-algorithm for SAT, so whenever the input Boolean formula F is unsatisfiable, every search ends up deciding F $\in \overline{\text{SAT}}$; and whenever F $\in$ SAT at least one search will answer F $\in$ SAT. If $f(A) \cap f(\overline{A}) \neq \emptyset$, then neither of the above two is true according to the adapted search procedure as presented in the proof of Theorem 2. By stipulating that $f(A) \cap f(\overline{A}) = \emptyset$ in the $\leq^{NP}_{dm}$ reduction, the possibility of premature pruning is eliminated. However, the sparse set of labels, which is used to guide the pruning of nodes with unsatisfiable formulas, might grow beyond sparseness, and thus results in a search that visits more than a polynomial amount of nodes, if too many labels in $\overline{S}$ were inadvertently added to it during the search. This undesirable possibility is remedied by using the counter to restrict the search to a polynomial size.

Before proceeding to the next theorem, we restate here some definitions from Meyer and Paterson [7]: A partial order $<$ on G^* is polynomially well-founded and length-related, OK for short, if and only if (a) every strictly decreasing chain is finite and there is a polynomial p such that every finite $<$-decreasing chain is shorter than p of the length of its maximum element, and (b) $x < y$ implies $|x| \leq q(|y|)$, for some polynomial q, and all x and y in G^*. A set A is self-reducible if and only if there is an OK partial order and an oracle Turing machine M such that M accepts A in polynomial time with oracle A and, moreover, on any input x in G^*, M asks its oracle only about words strictly less than x in the partial order. A self-reduction of A is conjunctive if for all $x \in G^*$, $x \in A$ if and only if $xi \in A$ for all i, where xi's are the smaller (with respect to the OK order) strings about which the self-reduction asks its oracle given input x.

Given the above definitions, it can be perceived now that Theorem 2 can still be strengthened to the following theorem which is simply the $\le^{NP}_{dm}$ version of Theorem 4 of [7], as the self-reduction of $\overline{SAT}$ is, of course, conjunctive.

Corollary 3: If L is conjunctively self-reducible and L $\le^{NP}_{dm}$ S, where S is a sparse set, then L ∈ NP.

Proof: Same as above proof of Theorem 2, except that the counter uses an appropriate polynomial which depends on L, S, and M, the NDTM transducer that reduces L to S.

Corollary 3 implies, for instance, that graph isomorphism is in NP ∩ coNP if the complement of it $\le^{NP}_{dm}$-reduces to a sparse set. Another consequence of Corollary 3 is the following corollary, which says that no NP algorithm can approximately solve a coNP-$\le^{NP}_{dm}$-hard problem closely enough unless NP = coNP. More precisely, define the class ANP (cf. [1,7]), read almost NP, as L ∈ ANP if and only if there is a polynomial p and a nondeterministic Turing machine M which accepts L such that the running time of M is bounded by p(n) on inputs of length n except for at most p(n) inputs of length n. Then we have:

Corollary 4: There is a coNP-$\le^{NP}_{dm}$-hard set in ANP iff NP = coNP.

Proof: We first show that (∃ A ∈ ANP) [L $\le^{NP}_{dm}$ A] ==> (∃ S) [S is sparse and L $\le^{NP}_{dm}$ S] (cf. [7]). Suppose M1 : L $\le^{NP}_{dm}$ A and M2 is an NDTM that accepts A and runs in polynomial time except for at most p(n) inputs of length n, for every n. Consider the following NDTM transducer M: Given input x, M first simulates M1 on x, if M1 produces no output on x, then M halts without output; if M1 on input x produces an output, say y, then M continues to simulate M2 on y for p(|y|) steps. If M2 accepts y within p(|y|) steps, then M outputs 'a' and halts, where 'a' is a fixed element of A; if M2 fails to accept y within p(|y|) steps in this simulation, then M outputs y and halts. It is clear that M : L $\le^{NP}_{dm}$ S, where S is a sparse set and S ⊆ A. Therefore if A is coNP-$\le^{NP}_{dm}$-hard and A ∈ ANP, then there exists a sparse set S such that $\overline{SAT}$ $\le^{NP}_{dm}$ S, implying NP = coNP by Theorem 2.

The nondeterministic version of the main Theorem 3 of [7] is the following:

Theorem 5: If L is self-reducible and L $\le^{NP}_{dm}$ S1, $\overline{L}$ $\le^{NP}_{dm}$ S2, where S1 and S2 are sparse sets, then L ∈ NP ∩ coNP.

Proof: Modify the proof of Theorem 3 of [7] as follows: Suppose M1 : L $\le^{NP}_{dm}$ S1, M2 : $\overline{L}$ $\le^{NP}_{dm}$ S2. To decide whether x ∈ L, start with two empty answer lists, list 1 and list 2, and conduct a depth-first search of the tree with root x. Whenever all the sons of a node, say y, have been searched (and whether y ∈ L consequently decided), then run both M1 and M2 on y. If Mi (i=1,2) halts with output, then enter the pair (Mi(y), answer(y)) into list i and continue the search; if both runs halt without outputs, then terminate the search and decide x ∉ L. When a node, say z, is first reached in the construction of the tree, run both M1 and M2 on z. If Mi produces an output which is already in list i, then abort the search below z and use the corresponding answer as the answer for z; otherwise simply continue the search. A counter is used to count the number of nodes searched, when this number exceeds p(.), an appropriate polynomial (which is determined from L, S1, S2, M1, and M2), terminate the search and decide $x \notin L$. Hence $L \in NP$, as the above procedure is essentially an NP algorithm for L. If, on the other hand, the above procedure decides $x \in L$ whenever the search is terminated due to either (a) no outputs of Mi's on a certain input, or (b) number of nodes searched exceeds limit, then it becomes an NP algorithm for $\bar{L}$. So $L \in NP \cap coNP$.

Corollary 6: If PSPACE is $\le_{dm}^{NP}$-reducible to a sparse set, then NP = PSPACE.

Proof: Suppose A is any self-reducible PSPACE-complete set, by assumption $A \le_{dm}^{NP} S$, where S is a sparse set. Then $\bar{A} \le_m^p A$, as PSPACE is closed under complementation, it follows $\bar{A} \le_{dm}^{NP} S$. So $A \in NP \cap coNP$, implying that NP = coNP = PSPACE.

4. Concluding Remarks

From the results of [10], it appears that the $\le_m^{NP}$-degree structure of Σ_2^p and the $\le_m^p$-degree structure of Σ_1^p (NP) are very much alike. This analogy motivated us to consider the question whether all known Σ_2^p-$\le_m^{NP}$-complete sets are one-one equivalent under nondeterministic reductions. We have therefore defined a reasonable concept of nondeterministic polynomial-time one-one reduction and found out that all known Σ_2^p-$\le_m^{NP}$-complete sets are indeed equivalent with respect to this reducibility. If all Σ_2^p-$\le_m^{NP}$-complete sets were $\le_1^{NP}$-equivalent, then NP $\neq$ coNP and no sparse sets can be Σ_2^p-$\le_m^{NP}$-complete. Investigating the consequence of sparse Σ_2^p-$\le_m^{NP}$-complete sets, we proved that many sparseness results relating $\le_m^p$ and P = NP? do carry over to $\le_{dm}^{NP}$ and NP = coNP?, e.g., the existence of a sparse Σ_2^p-$\le_{dm}^{NP}$-complete set implies that NP = coNP, where $\le_{dm}^{NP}$ is a restricted form of $\le_m^{NP}$.

An open question which is of theoretical interest in its own right is whether $\le_{dm}^{NP}$ and $\le_m^{NP}$ are equivalent. It is easily observed that the direct composition of two $\le_{dm}^{NP}$ reductions does not make a $\le_{dm}^{NP}$ reduction. On the other hand, the class NP - $\{\emptyset\}$ forms a single $\le_{dm}^{NP}$-degree, as well as a single $\le_m^{NP}$-degree. Also open are the following questions: (1) Is it true that the existence of sparse Σ_2^p-$\le_m^{NP}$-complete sets collapse some higher classes, e.g., $\Sigma_2^p = \Pi_2^p$? (2) If there exists a cosparse coNP-$\le_{dm}^{NP}$-hard set, then NP = coNP?

References

[1] Berman, P. Relationship between density and deterministic complexity of NP-complete languages. 5th International Colloquium on Automata, Languages and Programming (1978) Lecture Notes in Computer Science 62 63-71.

[2] Berman, L., and Hartmanis, J. On isomorphism and density of NP and other complete sets. SIAM J. Computing 6 (1977) 305-322.

[3] Cook, S.A. The complexity of theorem proving procedures. Proc. 3rd Annual ACM Symposium on Theory of Computing (1971) 151-158.

[4] Fortune, S. A note on sparse complete sets. SIAM J. Computing 8 (1979) 431-433.

[5] Long, T.J. Strong nondeterministic polynomial time reducibilities. Theoretical Computer Science 21 (1982) 1-25.

[6] Mahaney, S.R. Sparse complete sets: solution of a conjecture of Berman and Hartmanis. J. Computer and System Sciences 25 (1982) 130-143.

[7] Meyer, A.R., and Paterson, M.S. With what frequency are apparently intractable problems difficult. MIT Tech Memo. 126 (1979).

[8] Papadimitriou, C.H., and Yannakakis, M. The complexity of facets (and some facets of complexity). J. Computer and System Sciences 28 (1984) 244-259.

[9] Stockmeyer, L.J. The polynomial-time hierarchy. Theoretical Computer Science 3 (1977) 1-22.

[10] Sung, H.L. On nondeterministic polynomial time reducibility, complexity class DP, and the difference hierarchies. TR 84/069, Dept. of Computer Science, State Univ. of New York at Stony Brook (1984)

[11] Ukkonen, E. Two results on polynomial time truth-table reductions to sparse sets. SIAM J. Computing 12 (1983) 580-587.

[12] Yap, C.K. Some consequences of non-uniform conditions on uniform classes. Theoretical Computer Science 26 (1983) 287-300.

[13] Yesha, Y. On certain polynomial-time truth-table reducibilities of complete sets to sparse sets. SIAM J. Computing 12 (1983) 411-425.

[14] Young, P. Some structural properties of polynomial reducibilities and sets in NP. Proc. 15th Annual ACM Symposium on Theory of Computing (1983) 392-401.

A THEORY OF CLOSED COVERS: TO DECIDE PROGRESS OF COMMUNICATING FINITE STATE MACHINES

MOHAMED G. GOUDA, LOUIS E. ROSIER AND HSU-CHUN YEN
Department of Computer Sciences, University of Texas at Austin, Texas 78712

1. INTRODUCTION

Recently, networks of communicating finite state machines have been used to model communication protocols. It has been shown that, many real protocols can be modeled as networks of two (or more) finite state machines that communicate by exchanging messages over two one directional, FIFO channels [1, 2, 7, 9, 10, 13, 14]. Using this kind of modeling, many protocol design errors such as deadlocks, unspecified receptions, etc., can be discovered.

One problem that seems to be important in protocol design is to determine, given a network, whether the network will progress indefinitely. Although this problem is undecidable in general [2], some algorithmic techniques have been successfully developed for special classes of networks [1, 7, 9, 13, 14]. So far, the most commonly used method to verify indefinite progress is *state exploration analysis* (or *reachability set analysis*) [10, 12]. Basically this involves generating all reachable states of a network, and then examining them one by one to check whether a nonprogressable state exists. This technique has two main shortcomings. First, the number of states can be very large. In this case, the technique is rather time-consuming and therefore, not suitable for most real applications. Second, only those networks with finite reachability sets can be analysed using this method.

To overcome the above two difficulties, a proof methodological technique has recently been developed [3], based on constructing a special subset of the network states, called a *closed cover*. As far as indefinite progressibility is concerned, it has been shown that the existence of a closed cover is necessary and sufficient to guarantee that the communication between two machines will progress indefinitely [3]. This technique has also been shown useful in proving other network properties, such as liveness [4], livelock-freedom [5], etc. As we will see later, using closed covers to prove network properties is, in some sense, analogous to using predicate assertions to prove the correctness of sequential programs. One can view a closed cover as an assertion (or predicate) imposed on states of a network such that, the assertion is true in a state iff the state is in the closed cover. In this way, a closed cover of a network is analogous to a loop-invariant of a sequential program. It is, however, undecidable, in general, to determine whether a closed cover exists. Therefore, in most cases, some intuition and/or *a priori* knowledge about the network is usually required in order to be able to construct a closed cover. This is not surprising since it is also the case that, complete knowledge of a sequential program is usually necessary in order to find loop-invariants.

In this paper, in order to broaden their applicability, we first generalize the definition of closed covers given in [3]. Then we are able to show that for a wide variety of networks, there exists a "nice" closed cover for each network whose communication will progress indefinitely. These "nice" closed covers, more importantly, can be generated systematically and hence, the entire verification procedure can be theoretically automated. Our approach is based on the observation that a closed cover can be viewed as a language over the set of messages allowable in the channels. As it turns out, the language containment property plays an important role in testing whether a set is a closed cover. From formal language theory [8], we have that the containment property is decidable for the following types of languages: (They will be referred to as "nice" languages.)

- finite languages,
- regular languages,
- bounded semilinear languages.

Now consider a class of networks for which "nice" closed covers exist iff closed covers exist. For such classes of networks we are able to design a checking procedure to determine indefinite progress. More importantly, we are able to show that, for a wide variety of networks a "nice" closed cover exists iff the communication can progress indefinitely. Consequently, the problem becomes decidable for such classes of networks. It is important to note that the decidability result for each class presented here was previously known. In fact, for most of the two communicating finite state machine network classes, for which indefinite progressability was previously shown to be decidable [2, 6, 9, 13], the decidability of indefinite progressability follows trivially as a result of our results. Hence, we feel our approach (using closed covers) is interesting because it provides a unified underlining theory. Furthermore, our technique also gives some idea of what a closed cover in each class may look like, and thus may be quite useful when constructing a proof (i.e. closed cover) of indefinite progressability.

The remainder of this paper is organized as follows. Networks of communicating finite state machines are defined in Section 2. In Section 3, we generalize the definition of closed covers (see [3] for more restricted cases) and study their properties from the theoretical point of view. Also in this section, a decision procedure for determining indefinite progress is given as well. In Section 4, we classify some FIFO networks based on their closed cover languages. Due to space limitations the proofs have been omitted from this version.

2. COMMUNICATING FINITE STATE MACHINES

A *communicating finite state machine* (CFSM) M is a directed labelled graph with two types of edges, namely, *sending edges* and *receiving edges*. A sending (receiving) edge is labelled send(g) (receive(g)) for some *message* g in a finite set G of messages. A node in M is further specified as an *initial node*. Let M and N be two CFSM's over the same message set G. The pair (M,N) is called a *FIFO network* of M and N. A *state* of a network (M,N) is a 4-tuple [v,w,x,y], where v and w are nodes in M and N, respectively, and x and y are two strings over G. Informally, a state [v,w,x,y] represents that M and N have reached nodes v, w respectively; while the input channels of M and N contain the message sequences x and y, respectively. Let $[v_0,w_0,E,E]$ denote the initial state of (M,N), where v_0 and w_0 are initial nodes of M and N respectively, and E denotes an empty channel.

Let s=[v,w,x,y] be a state of network (M,N), and let e be an outgoing edge of v or w. A state s' is said to *follow* s over e iff one of the following conditions is satisfied: (s' is said to be a *following state* of s.)

1. e is labelled send(g), from v to v' in M, and s'=[v',w,x,y'], where y'=y.g, ("." is the standard string concatenation operator).

2. e is labelled send(g), from w to w' in N, and s'=[v,w',x',y], where x'=x.g.

3. e is labelled receive(g), from v to v' in M, and s'=[v',w,x',y], where x=g.x'.

4. e is labelled receive(g), from w to w' in N, and s'=[v,w',x,y'], where y=g.y'.

Let s and s' be two states of (M,N); s' is said to be *reachable from* s over edges $e_1,...,e_r$ $(r \geq 0)$ iff s=s' (r=0) or there exist states $s_0,s_1,...,s_r$ (r>0) such that, $s=s_0$, $s'=s_r$ and s_{i+1} follows s_i over e_{i+1}, for i=1,...,r-1. A *path* in M (N), starting in node v and ending in node v', is a sequence of edges $e_1,e_2,...,e_k$ in M (N) such that, e_1 is an outgoing edge of v, e_k is a incoming edge of v', and for every $1<i<k$, there exists some node v_i in M (or N) so that e_i and e_{i+1} are incoming and outgoing edges of v_i, respectively. Let p(v,v') denote such a path. Let p and q be two paths in M and N, respectively. An *interleaving* of p and q, denoted by (p+q), is a sequence of edges such that, an edge is in (p+q) iff it is in p or q, and furthermore, those edges belonging to p (or q) in (p+q) preserve the same order as in p (or q). A state s' is said to be *reachable from* a state s *via* two paths p and q in M and N respectively, denoted by s -p,q→ s', iff there exists some interleaving (p+q) such that s' is reachable from s over (p+q). A state is *reachable* in (M,N) iff it is reachable from the initial state of (M,N) via some path.

A state s is called a *blocking state* for network (M,N) iff s has no following state. A network is said to be able to *progress indefinitely* iff there is no reachable blocking state.

3. CLOSED COVERS

Let M=(V,E) be a communicating finite state machine, where V and E represent the sets of nodes and labelled edges, respectively. We define a *cut* of M , denoted by C_M, to be a set of nodes $\{v_1,...,v_k\}$ such that, for every directed cycle *l* in M, there exists some common node in both *l* and C_M. Define AM to be the directed graph constructed from M such that, every node i in C_M is partitioned into two nodes i_1 and i_2 where i_1 has all the output edges of node i and and i_2 has all the input edges of i. Node i_1 is called the *input version* of node i; and i_2 is called the *output version* of node i. A path $a_1 \rightarrow a_2 \rightarrow ... \rightarrow a_{n-1} \rightarrow a_n$ is called a *segment* iff a_1 and a_n, but no other intermediate node, are in C_M. Associated with each v in C_M, we define AM(v) to be a subgraph that contains all segments with starting node v. Since C_M breaks every cycle in M, it should be reasonably easy to see that AM and AM(v), for every v in C_M, are acyclic.

A *cut* of a network (M,N) is a pair (C_M,C_N) such that, C_M and C_N are *cuts* of M and N, respectively. Let p: $v_1 \rightarrow v_2$ and q: $w_1 \rightarrow w_2$ be two segments in M and N respectively. A state $[v_1,w_1,x_1,y_1]$ is called *proper* with respect to p and q, iff one of the following is true:

1. $[v_1,w_1,x_1,y_1]$ -p,q→ $[v_2,w_2,x_2,y_2]$ for some x_2 and y_2.

2. There is no path p': $v_1 \rightarrow v'$ (q': $w_1 \rightarrow w'$) such that p (q) is a proper prefix of p' (q') and $[v_1,w_1,x_1,y_1]$ -p,q'→ $[v_2,w',x',y']$ ($[v_1,w_1,x_1,y_1]$ -p',q→

$[v',w_2,x',y']$), for some x' and y'.

A state is called *proper* iff it is proper with respect to every pair of outgoing segments in M and N.

Let C be a set of, possibly infinite, states. C is called a *cover* of network (M,N) *over* the cut (C_M,C_N) iff for every state $[v_i,w_i,x_i,y_i]$ in C, $v_i \in C_M$ and $w_i \in C_N$. A *Closed Cover* C is a minimum cover (i.e. no redundent states) which satisfies the following conditions:

(1) $[v_0,w_0,E,E]$ is in C.

(2) For every s in C, s is proper.

(3) For every s=[v,w,x,y] in C, let p and q be two paths in the graphs AM(v) and AN(w) respectively.

If

(a) $[v,w,x,y] \text{ -p,q}\rightarrow [v',w',x',y']$ and

(b) no other state in AM(v) and AN(w) is reachable from [v',w',x',y'],

then

(i) v' and w' must be in the output versions of some nodes in M and N , respectively and

(ii) [v',w',x',y'] is in C.

Informally, the word "closed" refers to the fact that, for an arbitrary state s in C and two segments p and q in M and N respectively, either (M,N) can make no progress in s along p and q, or s -p,q→ s', for some s' in C. Let $C'=\{ s' \mid \exists\, s \in C, \exists$ segments p and q, such that $s \text{ -p,q}\rightarrow s' \}$. Clearly then, the (3) condition above requires that $C' \subseteq C$. The reader, at this point, should note the importance of the containment ($\subseteq$) property in using closed cover methods. Further details will be clarified in subsequent sections. Furthermore, there is a strong analogy between the closed cover method and the assertion method commonly used for proving the correctness of sequential programs. To see this, one can think of a closed cover C as a predicate P such that, for every state s, P(s) is true iff $s \in C$. Note also that, a cut that breaks the computation of a network into acyclic graphs corresponds to breaking loops in a sequential program. In other words, a cut of a network is analogous to the positions of loop-invariants applied to a sequential program. Finally, the condition that a closed cover must be "closed" corresponds to the condition that the predicate (assertion) must be true each time it is reached in a sequential program.

Next, we present a theorem which can be viewed as the foundation of the closed cover technique.

Theorem 3.1: A (FIFO) network can progress indefinitely iff there exists a closed cover.

In what follows, our intention is to investigate the theoretical foundation of the closed cover method. Our approach is to establish a connection between networks and languages. By doing this, we able to utilize results in formal language theory to cope with problems in networks. Furthermore, this kind of approach allows us to present a unified technique to verify network properties.

Given a closed cover C of a network (M,N), we define $L(C)=\{v\$w\#x\%y \mid [v,w,x,y] \in C\}$. Without loss of generality, we assume that v, w, x and y are over disjoint alphabets, and **\$**, #, % are special symbols. Let $\mathcal{S}$ be a class of languages. A closed cover C is said to be in $\mathcal{S}$ if $L(C) \in \mathcal{S}$. The following lemma shows that, in general, L(C) set can be as "ugly" as any recursive enumerable (r.e., for short) set.

Theorem 3.2: For an arbitrary r.e. set L, one can effectively construct a network (M,N) and a closed cover C such that, L=L(C).

Immediately, we have the following corollary:

Corollary 3.1: Given a network (M,N) and a set of states C, the question "Is C a closed cover of the network (M,N) ?", in general, is undecidable.

Although Theorem 3.2 seems to indicate a major drawback for using closed covers in a decision procedure, we are able to show that for a wide variety of networks, closed covers can be effectively tested. In what follows, for convenience we view a state [v,w,x,y] as a string v**\$**w#x%y, where v, w, x and y are over different alphabets.

We say that a family of languages $\mathcal{S}=\{s_1,s_2,\ldots\}$ is *verifiable* iff

1. $\mathcal{S}$ is r.e., and

2. the containment property is decidable in $\mathcal{S}$, i.e. given s_i and s_j in $\mathcal{S}$, the question "Is $s_i \subseteq s_j$?" is decidable.

Without loss of generality, here we only consider those languages that contain strings of the form v**\$**w#x%y. Given a network (M,N) and a cut $CT=(C_M,C_N)$. Let s be a language. We define

CT(s)=

$\{[v',w',x',y'] \mid \exists\, [v,w,x,y] \in s,\ p{:}v \rightarrow v'$ and $q{:}w \rightarrow w'$ are paths in AM(v) and AN(w) respectively, such that $[v,w,x,y] \xrightarrow{p,q} [v',w',x',y']$ and no other state in AM(v) and AN(w) is reachable from $[v',w',x',y']\}$, if $\forall\, [v,w,x,y] \in s$, $v \in C_M$ and $w \in C_N$.

undefined, otherwise.

A class of networks $\Im$ is called $\mathcal{S}$*_verifiable,* where $\mathcal{S}$ is a verifiable family of languages, iff for every (M,N) (in $\Im$) that can progress indefinitely, there exist a closed cover C over a cut $CT=(C_M,C_N)$ such that,

1. $C \in \mathcal{S}$,

2. $\forall\, s \in \mathcal{S}$, $CT(s) \in \mathcal{S}$.

Then the following procedure can be used to decide whether a given network (M,N) in $\Im$ has a closed cover:

Algorithm:

```
1. Generate $s_i$'s in sequence (i.e. i=1, 2, 3,...).
2. For all possible cuts $CT=(C_M,C_N)$.
    i. Compute $CT(s_i)$.
    ii. Return "yes" if $CT(s_i) \subseteq s_i$.
```

Since S is r.e., the above algorithm will eventually terminate iff (M,N) has a closed cover. Consequently, we have the following theorem:

Theorem 3.3: The indefinite progressability problem is decidable for verifiable networks.

Proof. The decision procedure consists of two algorithms running in parallel. One of them is used to search for a blocking state; while the other one is to find a closed cover. Clearly, one of the two algorithms must terminate eventually. So, the indefinite progressability problem is decidable for all verifiable networks. □

4. NETWORKS WITH NICE CLOSED COVERS

In this section, our main concern is to classify networks based on their "verifibility". Before going into detail, we need the following definitions. Let N denote the set of nonnegative integers and N^r be the set of r-dimensional vectors over N. For anr vector $v_0 \in N^r$ and any set $P\ (=\{v_1,...,v_m\}) \subset N^r$, the set $L(v_0,P)= \{x \mid \exists\ k_1,...,k_m \in N^r$ and $x=v_0+\Sigma_{i=1}^{m} k_i v_i\}$ is called a *linear set* over the *generator vectors* $v_0,v_1,...,v_m$. Any finite union of linear sets is called a *semilinear set*.

Let $E=\{a_1,...,a_n\}$ be a set of symbols. A language $L \subseteq E^*$ is *bounded* iff $L \subseteq a_1^* a_2^* ... a_n^*$. We define $\#(L)$ by: $\#(L)=\{(i_1,...,i_n) | a_1^{i_1}...a_n^{i_n} \in L\}$, which is usually called the *Parikh* map of L. A language is called *bounded semilinear* iff it is over bounded languages and its parikh map is semilinear.

Here, we have a well known lemma, which serves an important role throughout the remainder of this section.

Lemma 4.1 : The following families of languages are verifiable:

1. finite languages (F),
2. regular languages (R),
3. bounded semilinear languages (BS).

In what follows, we show that for several restricted classes of networks, some closed cover (for each indefinitely progressible network) falls into one of the above categories. Consequently, given a network in those classes, a decision procedure exists for determining whether such a network can progress indefinitely. Now the indefinite progressablity problem was previously shown to be decidable for each of the networks presented here

[2, 6, 9, 13]. Our work is merely provides a unifying theory. An important additional motivation, however, is that while decision procedures exist, many times in practice they are intractable. Thus, instead of algorithmically determining the given property, the user is faced with "proving" that the network can progress indefinitely. Our results indicate for several network classes what form a closed cover (if one exists) can be expected to take.

4.1 Networks with finite closed covers

In this subsection, we consider the following three classes of networks:

C_1: Networks with bounded communication (i.e. networks with finite reachability sets).

C_2: Networks with one channel bounded [2, 9].

C_3: Networks with bounded fair reachability graphs [13].

Now, for any network (M,N) in one of these three classes, consider a cut (C_M, C_N) in such a way that, C_M (C_N) contains every nodes in M (N). We define $C=\{[v,w,x,y] \mid [v,w,x,y]$ is a reachable state and $|x|=|y|\}$. It can be shown that C, indeed, is a closed cover as long as (M,N) can progress indefinitely. Moreover, as discussed in [13], C is exactly the set of fair reachability states, which can easily be shown to be finite for classes C_1, C_2 and C_3 networks. Consequently, we have the following result:

Theorem 4.1: C_1, C_2 or C_3 are all F_verifiable.

4.2 Networks with regular closed covers

Here we consider the following class of networks:

C_4: Networks of two CFSM's, such that at least one machine, with no sending loop, sends only one type of message to the other.

Let (M,N) be a network in C_4, and without loss of generality, we assume that M sends only one type of message and has no sending loop. In what follows, we say two networks N_1 and N_2 are *equivalent* if it is the case that N_1 can progress indefinitely iff N_2 can. To use the technique of closed covers, we require the following lemmas:

Lemma 4.2: For an arbitrary network (M,N), one can effectively construct an equivalent network (M",N") such that each node in M" (or N") has all its incoming edges labelled the same.

Using this lemma, one can convert each network in C_4 into an equivalent network consisting of only one type of incoming edge (either sending or receiving, but not both). Let C'_4 be the result of this kind of transformation. We then have:

Lemma 4.3: For an arbitrary network (M,N) in C'_4, one can effectively construct an equivalent network (M',N') in C'_4, in which N' has no receiving loop.

Given a network (M,N) in C_4, let (M',N') be an equivalent network that satisfies those conditions in Lemmas 4.2 and 4.3. We let $C_{M'}=\{n \mid n$ is a node in M' with incoming edges labelled with **receive**} and $C_{N'}$={n|n is a node in N' with incoming edges labelled with **send**}. We further define C'={[v,w,x,y]|[v,w,x,y] is reachable by following the computation of the cut $(C_{M'},C_{N'})$}. Then we have:

Lemma 4.4: C' is a closed cover iff (M,N) will progress indefinitely.

Lemma 4.5: C' is regular.

By Lemmas 4.4 and 4.5, we immediately have:

Theorem 4.2: C_4 is R_verifiable.

4.3 Networks with bounded semilinear closed covers

Finally, we turn our attention to some networks that have more complicated closed covers, i.e., bounded semilinear closed covers. Consider the following two classes of networks:

C_5: Networks of two CFSM's that exchange only one type of message [13].

C_6: Networks of two CFSM's such that one machine, with no sending loop, sends a bounded language over its output channel [6]. (This is a generalization of the class C_4)

We can show that:

Theorem 4.3: C_5 and C_6 are BS_verifiable.

Following Theorems 3.3, 4.1, 4.2 and 4.3, we have:

Theorem 4.4: The indefinite progressability problem is decidable for classes C_1 to C_6 networks.

Furthermore, we know that when trying to prove the indefinite progressability problem using the closed cover method (for networks in C_1-C_6), it is sufficient to only consider closed covers which are finite (for C_1-C_3), regular (for C_4) , or bounded semilinear (for C_5-C_6).

REFERENCES

[1] Bochmann, G., Finite State Description of Communication Protocols, *Computer Networks*, Vol. 2, 1978, pp. 361-371.

[2] Brand, D. and Zafiropulo, P., On Communicating Finite State Machines, *JACM,* Vol. 30, No. 2, April 1983, pp. 361-371.

[3] Gouda, M., Closed Covers: To Verify Progress for Communicating Finite State Machines, *IEEE Trans. on Software Eng.*, Vol. SE-10, No. 6, Nov. 1984, pp. 846-855.

[4] Gouda, M. and Chang, K., A Technique for Proving Liveness of Communicating Finite State Machines with Examples, *Third ACM Symposium on Principles of Distributed Computing,* August 1984.

[5] Gouda, M., Chow, C. and Lam, S., Livelock Detection in Networks of Communicating Finite State Machines, Tech. Rep. No. 84-10, Univ. of Texas at Austin, Dept. of Computer Sciences, March 1984.

[6] Gouda, M., Gurari, E., Lai, T. and Rosier, L., On Deadlock Detection in Systems of Communicating Finite State Machines, Tech. Rep. No. 84-11, Univ. of Texas at Austin, Dept. of Computer Sciences, 1984. Revised April 1985.

[7] Gouda, M. and Rosier, L., Priority Networks of Communicating Finite State Machines, *SIAM Journal on Computing*, Vol. 14, No. 3, August 1985, pp. 569-584.

[8] Hopcroft, J. and Ullman, J., "Introduction to Automata Theory, Languages, and Computation", Addison-Wesley, Reading, Mass., 1979.

[9] Rosier, L. and Gouda, M., On Deciding Progress for a Class of Communicating Protocols, *Proceedings of the Eighteen Annual Conference on Information Sciences and Systems*, Princeton Univ., 1984, pp. 663-667.

[10] Sunshine, C., Interprocess Communiction Protocols for Computer Networks, *Ph.D dissertation*, Dept. of Computer Sciences, Stanford Univ., 1975.

[11] Sunshine, C., Formal Modeling of Communication Protocols, USC/Inform. Sc. Institute, Research Report 81-89, March 1981.

[12] West, C., An Automated Technique of Communication Protocol Validation, *IEEE Trans. on Comm.*, Vol. COM-26, August 1978, pp. 1271-1275.

[13] Yu, Y. and Gouda, M., Deadlock Detection for a Class of Communicating Finite State Machines, *IEEE Trans. on Comm.*, Vol. COM-30, No. 12, December 1982, pp. 2514-2518.

[14] Yu, Y. and Gouda, M., Unboundedness Detection for a Class of Communicating Finite State Machines, *Information Processing Letters*, 17, December 1983, pp. 235-240.

AN EVENT ALGEBRA FOR THE STUDY OF LIVENESS IN SELF-TIMED COMPUTATIONAL NETWORKS *)

RAMI MELHEM **)
The University of Pittsburgh
Pittsburgh, PA. 15260.

ABSTRACT

A class of data-driven computational networks is defined where each cell does not read any of its inputs unless all inputs are available and does not write any of its outputs unless previous results are consumed. An algebra of events is then developed to express execution histories of such networks and to prove that liveness depends uniquely on the initial state of the network. Moreover, it is proved that a network is deadlock-free if and only if each cell completes the execution of at least one read/write cycle. This means that only the initial segment of any simulation may provide a proof of liveness. Finally, sufficient and necessary conditions are derived for the verification of liveness without any simulation.

1. INTRODUCTION

Systolic [4] and self-timed [11] synchronizations are two different approaches for the organization of large VLSI computational networks. In the systolic organization, a global clock is used for the synchronization of all the cells in the network, while synchronization in self-timed systems is achieved by local protocols between neighboring cells.

Self-timed techniques may be easy to apply on pipelined systems where data flows in one direction. However, it seems that little attention has been paid to the complications that may result from the application of these techniques to networks with data flowing in more than one direction. More specifically, a network may deadlock if each cell is waiting to receive its input from other cells.

Many methods were introduced in the literature for the verification of temporal properties in communicating processes (e.g. [1] ,[3]) and in asynchronous systems (e.g. [10]). However, these techniques do not seem adequate for the type of results that we are seeking. More specifically, we want to prove that deadlock in self-timed computational networks depends uniquely on the initial state, and then identify the initial states that guarantee deadlock-free computation. For this, we develop an algebra of events, where the execution history of a computation is represented by a sequence of events, and the state of the network, at any time, is represented by a state matrix that is modified after the occurrence of each event according to some axioms. This algebra is used to study the properties of execution histories and their dependence on the initial state of the network. Similar algebras of events were introduced in [2] and [6] but used in different contexts.

In this paper, we consider a class of self-timed networks where each cell does not start its computation before all its inputs are available and does not output the results before all previous outputs are consumed. The application of the algebra of events to other types of self-timed protocols requires only minor modifications.

---

*) This work was, in part, supported under ONR contract N00014-80-C-0455.

**) On leave from Purdue University

Finally, we note that the model presented in [8] for the verification and simulation [9] of systolic networks may be extended to include self-timed networks only if the network is proven to be deadlock-free [7].

2. A CLASS OF SELF-TIMED NETWORKS

A self-timed network is defined here as a set of cells, each having a certain number of input and/or output ports, and a set of unidirectional communication links, each connecting an output port of some cell to an input port of another cell.

A communication link $l_{i,j}$ directed from an output port of a cell c_i to an input port of another cell c_j is a medium that carries always one data item of a certain type. At any instant, this medium may be in one of two states, namely RE (read enabled) or WE (write enabled). At the beginning of the operation of the network, $l_{i,j}$ may be either initially read enabled (IRE) or initially write enabled (IWE). We will use the usual convention of calling $l_{i,j}$ an output link of c_i and an input link to c_j. We will also assume that the networks do not contain direct feed back links of the form $l_{i,i}$. This type of feed back may be replaced by internal registers.

The cells in a computational network may be classified into three classes; namely 1) input cells that have only output ports, 2) output cells that have only input ports, and 3) computational cells that have both input and output ports. Each cell in the network repeats, indefinitely, the execution of the following cycle.

Algorithm CYCLE

```
1)  1.1. Wait until all input links are in state RE ;
    1.2. Read from each input port (if any) one data item ;
    1.3. Reset the state of the input links to WE ;
2)  Case (cell-type) of
    Computational   : Compute a data item for each output port ;
    Input           : Create a data item for each output port ;
    Output          : Inhale the data items read in step 1 ;
3)  3.1. Wait until all output links are in state WE ;
    3.2. Write on each output port (if any), the corresponding item from step 2.
    3.3. Reset the states of the output links to RE ;
```

In order to obtain a useful computational model, we assume that the execution of step 2 in CYCLE always terminates. This, in particular, assumes that a data item is eventually supplied whenever the network is waiting for an input and that the output of the network is eventually consumed. We also assume that at most one link may be directed from a specific cell i to another cell j.

Let N be a self-timed network composed of n cells labeled by the integers $1, \cdots, n$ and let the topology of N be specified by the adjacency matrix A defined such that $A_{i,j} = 1$ if there is a link $l_{i,j}$ directed from cell i to cell j, and $A_{i,j} = 0$, otherwise. Given that during the execution of N, a link may be in one of two states, we express the states of the n links in terms of an $n \times n$ matrix S defined by:

$$S_{i,j} = \begin{cases} 0 & \text{if } A_{i,j} = 0 \\ 1 & \text{if } A_{i,j} = 1 \text{ and } l_{i,j} \text{ is } RE \\ -1 & \text{if } A_{i,j} = 1 \text{ and } l_{i,j} \text{ is } WE \end{cases} \tag{1}$$

The different steps in CYCLE provide a convenient way for describing the state of a certain cell during the execution of N. More specifically, a cell is said to be in state R (reading), C (computing) or W (writing) if it is executing steps 1, 2 or 3 of CYCLE, respectively. Noting that direct loops are not allowed in N, the diagonal elements of S may be used to specify the states of the n cells. More specifically, we set

$$S_{i,i} = \begin{cases} 1 & \text{if } cell\ i\ is\ in\ state\ R \\ 0 & \text{if } cell\ i\ is\ in\ state\ C \\ -1 & \text{if } cell\ i\ is\ in\ state\ W \end{cases} \tag{2}$$

The matrix S defined by (1) and (2) is called the state matrix.

3. EVENTS AND INSTANCES OF EVENTS

Given a particular cell k in a self-timed network, we refer to any change in the state of that cell by an "event". More specifically, the events $E_{r,k}$, $E_{c,k}$ and $E_{w,k}$ are defined as the events of leaving state R, leaving state C, and leaving state W, respectively. We also define the t^{th} instance $E_{x,k}^{(t)}$ of an event $E_{x,k}$, $x = r,c$ or w, as the occurrence of $E_{x,k}$ during the execution of the t^{th} cycle of cell k.

The states of the links connected to cell k play a key role in the occurrence of event instances in that cell. For example, if cell k is executing step 1.1 in its t^{th} cycle, then it may not terminate the execution of this step unless all its input links are RE. That is the occurrence of $E_{r,k}^{(t)}$, for any t, is feasible only if cell k is in state R and all its input links are RE. Similarly, conditions may be given for the feasibility of $E_{w,k}$ and $E_{c,k}$. They are formalized by the following definitions:

Definition 1: Let A be the adjacency matrix of a network N composed of n cells. The domain of any event $E_{x,k}$, $1 \le k \le n$ is defined as follows:

$$dom(E_{x,k}) = \begin{cases} \{(j,k) \mid 1 \le j \le n,\ A_{j,k} = 1\} \cup \{(k,k)\} & \text{if } x = r \\ \{(k,k)\} & \text{if } x = c \\ \{(k,j) \mid 1 \le j \le n,\ A_{k,j} = 1\} \cup \{(k,k)\} & \text{if } x = w \end{cases}$$

Definition 2: Given a state S of N, an event $E_{x,k}$ is said to be feasible at S if:

For x = r : $S_{i,j} = 1 \qquad (i,j)\ \varepsilon\ dom(E_{x,k})$

For x = c : $S_{i,j} = 0 \qquad (i,j)\ \varepsilon\ dom(E_{x,k})$

For x = w : $S_{i,j} = -1 \qquad (i,j)\ \varepsilon\ dom(E_{x,k})$

Definition 3: A state S is called a dead state if no events are feasible at S. []

Clearly, the occurrence of an event instance changes the state of the network. More specifically, for any positive integer t, The occurrence of $E_{r,k}^{(t)}$ changes the state of cell k from R to C, and sets the states of its input links to WE. The changes caused by the occurrence of $E_{w,k}^{(t)}$ or $E_{c,k}^{(t)}$ are similar. They are described by the following axiom.

State transformation Axiom : Let $E_{x,k}$ be an event feasible at state S of N, and let $\bar{S}$ be the state of N after the occurrence of an instance $E_{x,k}^{(t)}$ of $E_{x,k}$. If $E_{x,k}^{(t)}$ is the only instance that occurred between S and $\bar{S}$, then

1) For $(i,j)\ \not\varepsilon\ dom(E_{x,k})$ $\qquad \bar{S}_{i,j} = S_{i,j}$

2) For $(i,j)\ \varepsilon\ dom(E_{x,k})$

if $x = r$ then $\qquad \bar{S}_{i,j} = \begin{cases} -1 & i \ne j \\ 0 & i = j \end{cases}$

if $x = c$ then $\qquad \bar{S}_{i,j} = -1$
if $x = w$ then $\qquad \bar{S}_{i,j} = 1$. []

4. EXECUTION HISTORY OF SELF-TIMED NETWORKS

An execution history of a self-timed network N is expressed as a sequence $H = V_1, V_2, \cdots$, where each V_i is a set that contains event instances. Every event instance that occurred during execution should appear in some V_i such that 1) each V_i contains instances that occurred simultaneously, and 2) V_{i-1} contains instances that occurred before those in V_i. Note that if N does not deadlock, then H is infinite.

Let S be some state of N. We will use the notation $S \xrightarrow{V} \bar{S}$, where V is a set of event instances, to indicate that each event in V is feasible at S and that $\bar{S}$ is the state of N after the simultaneous occurrence of the instances in V.

Given an initial state S^0 for N, different runs of N may result in different execution histories, depending on the time that each cell spends in the different steps of its cycles. However, any particular execution history, say $H = V_1, V_2, \cdots$, should satisfy the following properties:

[P1] There exists a sequence of states $S^0, S^1, \cdots$, associated with H such that for any $i > 0$, $S^{i-1} \xrightarrow{V_i} S^i$. That is, at any instant during execution, only instances of feasible events may occur.

[P2] An instance $E_{x,k}^{(t)}$, for some $t \geq 1$, $1 \leq k \leq n$ and $x = r, c$ or w, may occur at most once in H. Moreover, if $t > 1$ and $E_{x,k}^{(t)} \in V_i$, for some i, then, there exists a $j < i$ such that $E_{x,k}^{(t-1)} \in V_j$. That is, the $(t-1)^{st}$ instance of an event should occur before its t^{th} instance.

[P3] If $H = V_1, \cdots, V_m$ is finite, then S^m is a dead state. That is, execution stops only if N deadlocks.

Definition 4: Any sequence of sets of event instances that satisfies the above three properties is called an S^0-feasible sequence. []

By definition, any execution history of N starting from S^0 is an S^0-feasible sequence. The mathematical definition of feasible sequences makes them easy to construct and study. The following lemmas and definitions are needed in order to prove some properties about feasible sequences, and thus execution histories.

Lemma 1: If $E_{x,k}$ and $E_{y,q}$ are two different events feasible at a certain state S of N, then

$$dom(E_{x,k}) \cap dom(E_{y,q}) = \phi \tag{3}$$

Proof : If $x \neq y$ then (3) follows directly from Definition 2. If $x = y$, then from Definition 1, (3) is false only if $k = q$, which contradicts the hypothesis that $E_{x,k}$ and $E_{y,q}$ are different. []

From the above lemma, it is clear that if $V = \{e^{(t)}, v^{(t)}\}$, and $S^a \xrightarrow{V} S^{a+1}$, then $dom(e) \cap dom(v) = \phi$ and hence S^{a+1} may be obtained from S^a by application of the state transformation axiom to e and v independently. The following lemma proves that if an event is feasible at a certain state, then it remains feasible until the occurrence of one of its instances.

Lemma 2: Let e be an event feasible at S^a, and let $S^a \xrightarrow{V_{a+1}} S^{a+1} \rightarrow \cdots \xrightarrow{V_m} S^m$. If $V_{a+1}, \cdots, V_m$ do not contain any instance of e, then e is feasible at S^m.

Proof: Let $v^{(t)} \in V_{a+1}$. Hence, v is feasible at S^a, and so is e (from the hypothesis). By Lemma 1, $dom(e) \cap dom(v) = \phi$, and from the state transformation axiom, the occurrence of $v^{(t)}$ does affect only the elements of S^a that are in $dom(v)$. This is valid for any $v^{(t)}$ in V_{a+1}, and hence, $S_{i,j}^{a+1} = S_{i,j}^a$ for $(i,j) \in dom(e)$, that is, e is feasible at S^{a+1}. The feasibility of e at S^m follows by repeating this argument $m-a$ times. []

Next, we redefine Lamport's partial ordering of events [5] to suit our model.

Definition 5: Two event instances I_1 and I_2 are related by the ordering $I_1 <_{S^0} I_2$ if for any S^0-feasible sequence $V_1, V_2, \cdots, I_2 \varepsilon V_i$, for some i implies that $I_1 \varepsilon V_j$, for some $j < i$. That is, if I_2 does occur, then I_1 should occur before I_2. The subscript S^0 will be dropped when it is obvious from the context. []

Lemma 3: Given an initial state S^0 for N, the following partial orderings do hold for any S^0-feasible sequence, and $t \geq 1$.

$$E_{r,i}^{(t)} < E_{c,i}^{(t)} < E_{w,i}^{(t)} < E_{r,i}^{(t+1)} \qquad \text{for } any\ i \tag{4a}$$

$$E_{r,j}^{(t)} < E_{w,i}^{(t)} < E_{r,j}^{(t+1)} \qquad \text{for } any\ i \neq j\ with\ S_{i,j}^0 = 1 \tag{4b}$$

$$E_{w,i}^{(t)} < E_{r,j}^{(t)} < E_{w,i}^{(t+1)} \qquad \text{for } any\ i \neq j\ with\ S_{i,j}^0 = -1 \tag{4c}$$

Proof: see [7].

Lemma 4: If $dom(e) \cap dom(v) \neq \phi$, then for any initial state S^0 and $t > 1$, there exists a $T \geq 1$ such that

$$e^{(T)} <_{S^0} v^{(t)}. \tag{5}$$

Proof: From Definition 1, the following are the only cases where the domains of two events do intersect:

1) $dom(E_{r,j}) \cap dom(E_{w,i}) = \{(i,j)\},\ i \neq j$

2) $dom(E_{x,i}) \cap dom(E_{y,i}) = \{(i,i)\},\ x,y = r, c\ or\ w$, and $x \neq y$.

The proof of (5) in the first case follows from (4b/c) and in the second case follows from (4a). []

Next, we prove that deadlock in a self-timed network depends uniquely on the initial state, irrespective of any particular execution history.

Theorem 1: Given an initial state S^0 of N, if there exists a finite S^0-feasible sequence, then any S^0-feasible sequence is finite. Conversely, if there exists an infinite S^0-feasible sequence, then any S^0-feasible sequence is infinite.

Proof: Let $V_1, V_2, \cdots, V_m$ be a finite S^0-feasible sequence such that $S^0 \xrightarrow{V_1} S^1 \cdots \xrightarrow{V_m} S^m$, and S^m is a dead state. Let also $U_1, U_2, \cdots$ be any other S^0-feasible sequence such that $S^0 \xrightarrow{U_1} \overline{S}^1 \xrightarrow{U_2} \overline{S}^2 \cdots$. We prove by induction that for any $k \geq 1$ and any $I \varepsilon U_k$, there exists a q such that $I \varepsilon V_q$.

For k = 1: Let $I \varepsilon U_1$ and assume that $I \not\varepsilon V_1 \cup \cdots \cup V_m$. I is feasible at S^0, and by Lemma 2 it is feasible also at S^m, which contradicts the hypothesis that S^m is dead. Hence $I \varepsilon V_q$ for some q.

For k > 1: Assume that every instance of event in $U_1, \cdots, U_{k-1}$ does occur in $V_1, \cdots, V_m$, and consider any $I \varepsilon U_k$, where I may be an instance of $E_{r,i}$, $E_{c,i}$ or $E_{w,i}$, for some i. We will prove the induction step only for $I = E_{r,i}^{(t)}$, for some t. The proof in the other two cases is similar. Let $R_i = \{(j,i) \mid (j,i) \varepsilon dom(E_{r,i}), j \neq i, S_{j,i}^0 = 1\}$, and $W_i = \{(j,i) \mid (j,i) \varepsilon dom(E_{r,i}), S_{j,i}^0 = -1\}$. First consider the case $t = 1$ and assume that $E_{r,i}^{(1)} \varepsilon U_k$ but $E_{r,i}^{(1)} \not\varepsilon V_1 \cup \cdots \cup V_m$. By Lemma 3, for each $(j,i) \varepsilon W_i$, $E_{w,j}^{(1)} < E_{r,i}^{(1)}$, that is $E_{w,j}^{(1)}$ does occur in $U_1, \cdots, U_{k-1}$, and, by the induction hypothesis, in $V_1, \cdots, V_m$. But $E_{w,j}^{(1)}$ sets the $(j,i)^{th}$ entry of the state matrix to 1, and only an instance of $E_{r,i}$ may change this entry. Given that $E_{r,i}^{(1)} \not\varepsilon V_1 \cup \cdots \cup V_m$, we conclude that

$S^m_{j,i} = 1$ for any $(j,i) \varepsilon W_i$. But for any $(j,i) \varepsilon R_i \bigcup \{(i,i)\}$, we have $S^0_{j,i} = 1$, and because $E^{(1)}_{r,i} \not\varepsilon V_q \bigcup \cdots \bigcup V_m$, we also have $S^m_{j,i} = 1$. Hence, for any $(j,i) \varepsilon dom(E_{r,i})$, $S^m_{j,i} = 1$, that is $E_{r,i}$ is feasible at S^m, which contradicts that S^m is dead. Hence, $E^{(1)}_{r,i} \varepsilon V_q$ for some q.
Now let $t > 1$, and assume that $E^{(t)}_{r,i} \varepsilon U_k$. By Lemma 3,
$E^{(t-1)}_{r,i} < E^{(t-1)}_{w,i} < E^{(t)}_{r,i}$,
$E^{(t-1)}_{r,i} < E^{(t-1)}_{w,j} < E^{(t)}_{r,i}$ for $(j,i) \varepsilon R_i$
$E^{(t-1)}_{r,i} < E^{(t)}_{w,j} < E^{(t)}_{r,i}$ for $(j,i) \varepsilon W_i$
Hence, for $(j,i) \varepsilon R_i \bigcup \{(i,i)\}$, $E^{(t-1)}_{w,j}$ does occur in $U_1, \cdots, U_{k-1}$ and by the induction hypothesis in $V_1, \cdots, V_m$. Similarly, for $(j,i) \varepsilon W_i$, $E^{(t)}_{w,j}$ does occur in $V_1, \cdots, V_m$. Each of these instances of events occurs after $E^{(t-1)}_{r,i}$ and sets the corresponding entry of the state matrix to 1. If we assume that $E^{(t)}_{r,i} \not\varepsilon V_q$, $q = 1, \cdots, m$, then $S^m_{j,i} = 1$ for any $(j,i) \varepsilon R_i \bigcup W_i \bigcup \{(i,i)\} = dom(E_{r,i})$, and $E_{r,i}$ is feasible at S^m. This contradicts that S^m is dead, and proves that $E^{(t)}_{r,i} \varepsilon V_q$ for some q, which completes the induction proof.

Now, $V_1, \cdots, V_m$ may contain only a finite number of instances of events, and each event may occur only once in a feasible sequence. Hence, $U_1, U_2, \cdots$ contains only a finite number of instances of events, which proves that it is finite. The second part of the theorem is a direct consequence of the first part. []

Definition 6: An initial state S^0 is live if there exists an infinite S^0-feasible sequence. Otherwise S^0 is non-live. []

By Theorem 1, any execution of N that starts at a live state may not deadlock, and any execution of N that starts at a non-live state will definitely deadlock.

5. DETERMINATION OF LIVE INITIAL STATES

Theorem 1 allows the use of any feasible sequence (or execution history) for the proof of liveness of initial states. Hence, we will consider sequences where events in the first cycle are always given a higher priority to occur. More specifically, we define "First Cycle First" (FCF) event sequences as follows:

Definition 7: An S^0-FCF sequence, $V_1, V_2, \cdots$, is constructed as follows:

1) $V_1 = E^{(1)}_{x,k}$, where $E_{x,k}$ is any event feasible at S^0.
2) For any $i \geq 1$, let $S^0 \xrightarrow{V_1} S^1 \rightarrow \cdots \xrightarrow{V_i} S^i$ and let F_{i+1} be the set of events feasible at S^i. If $F_{i+1} = \phi$, then terminate the construction, else if there exists an event $E_{y,q} \varepsilon F_{i+1}$ such that $E^{(1)}_{y,q} \not\varepsilon V_1 \bigcup \cdots \bigcup V_i$, then take $V_{i+1} = \{E^{(1)}_{y,q}\}$. Otherwise, set $V_{i+1} = \{E^{(t)}_{z,p}\}$ for any $E_{z,p} \varepsilon F_{i+1}$, where t is such that $E^{(t-1)}_{z,p}$ but not $E^{(t)}_{z,p}$ is in $V_1, \cdots, V_i$. []

S^0-FCF sequences are, by construction, S^0-feasible, and hence they may be used to determine the liveness of S^0.

Theorem 2: Let $V_1, V_2, \cdots$ be any S^0-FCF sequence and let m be the smallest integer such that $V_m = \{E^{(t)}_{x,k}\}$ for some x, k and $t \neq 1$. Then S^0 is live if and only if $m = 3n$, where n is the number of cells in the corresponding network.

Proof: Let $S^0 \xrightarrow{V_1} S^1 \xrightarrow{V_2} \cdots$, and let F_i be the set of events feasible at S^i. First, note that $m \leq 3n$ because only $3n$ events are defined, namely $E_{x,k}$ for $x = r, c, w$ and $k = 1, \cdots, n$. If $m = 3n$, then the first instance of each of these $3n$ events occurs, and it is easy to show that, irrespective of the order of occurrence of these events, we have $S^{3n} = S^0$. Now, let $V^t_i = \{E^{(t)}_{x,k}\}$, where $V_i = \{E^{(1)}_{x,k}\}$, and construct the sequence $V_1, \cdots, V_{3n}, V^2_1, \cdots, V^2_{3n}, V^3_1, \cdots$. This sequence is S^0-feasible and infinite, which proves, by theorem 1, that S^0 is live.

Now, if $m < 3n$, then there exists a non empty set of events A such that for each $e \varepsilon A$, $e^{(1)} \not\varepsilon V_1 \cup \cdots \cup V_m$, and $e \not\varepsilon F_m$. Let e be any event in A. For any $v^{(t)} \varepsilon V_{m+1}$, $t > 1$, we have by Lemma 4 $dom(e) \cap dom(v) = \phi$, because otherwise some instance of e would have occurred before $v^{(t)}$. The empty intersection of $dom(e)$ and $dom(v)$ implies that the occurrence of v will not change any entry in $dom(e)$, and hence e is not feasible at S^{m+1}. By repeating this argument, we may prove that e is never feasible, and, thus, never occurs in $V_1, V_2, \cdots$.

Let $\alpha = \{i \mid E_{x,i} \varepsilon A \text{ for some } x = r, c \text{ or } w\}$. For any $i \varepsilon \alpha$, Lemma 3 implies that $E_{w,i}^{(1)}$ does not occur in $V_1, V_2, \cdots$, because this would require that all of $E_{r,i}^{(1)}$, $E_{c,i}^{(1)}$ and $E_{w,i}^{(1)}$ do occur in $V_1, V_2, \cdots$, which contradicts the definition of α. Clearly, α constitute the set of cells that do not complete the first cycle.

Next, we will prove that any event $E_{y,j}$, $j \not\varepsilon \alpha$, does occur in $V_1, V_2, \cdots$ only finitely many times. Pick some $i \varepsilon \alpha$. From the connectivity of the network, there is either a path form cell j to cell i or a path from cell i to cell j. If the first case applies, then by repeatedly applying Lemma 3, we can find a τ such that $E_{r,i}^{(2)} < E_{w,j}^{(\tau)}$. But $E_{r,i}^{(2)}$ does no occur in $V_1, V_2, \cdots$, then $E_{w,j}^{(\tau)}$ and consequently $E_{r,j}^{(t)}$, $E_{c,j}^{(t)}$ and $E_{w,j}^{(t)}$ for $t > \tau$, do not occur in $V_1, V_2, \cdots$. Similarly, if there is a path from cell i to cell j, then, we may find a σ such that $E_{w,i}^{(2)} < E_{r,j}^{(\sigma)}$, and prove that $E_{r,j}^{(t)}$, $E_{w,j}^{(t)}$ and $E_{c,j}^{(t)}$, for $t > \sigma$, do not occur in $V_1, V_2, \cdots$.

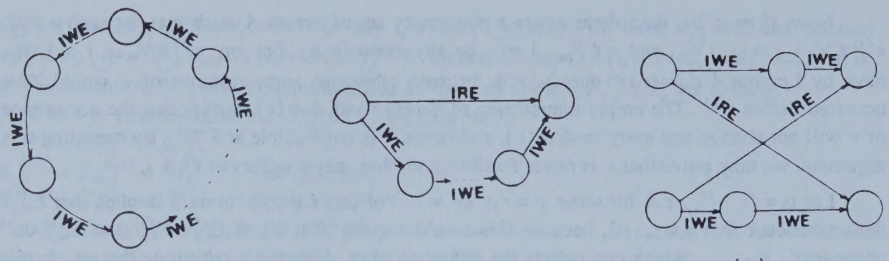
Hence, any event may only occur in the S^0-feasible sequence $V_1, V_2, \cdots$ finitely many times, which by Theorem 1 proves that S^0 is not a live state. []

By applying Theorem 2, we may test the liveness of a given initial state S^0 of a network N by simulating an execution history where only the first cycle (3 events) is allowed to execute in any cell. If in this execution every cell is able to complete its first cycle, then S^0 is live, otherwise it is not. In [7], we give an algorithm that simulates the execution of the first m instances of an S^0-FCF sequence. More specifically, at any particular state, feasible events are identified and instances of these events are executed causing the appropriate state transformations. However, only the first instance of any event is allowed to execute, and The simulation terminates when no events are feasible. The initial state is then declared live if every cell terminates the execution of its first cycle. Let **LIVE** be the name of this algorithm.

Next, we would like to determine the liveness of an initial state by a static analysis, that is without any simulation. This analysis may be based on the properties of the matrices corresponding to live initial states. For example, it is easy to show that if the elements of the matrix S^0 are non negative (all links are IRE), then LIVE declares S^0 live in one iteration. Also, if $S_{i,j}^0 \neq 1$ for $i < j$, and $S_{i,j}^0 \neq -1$ if $i > j$, Then LIVE declares S^0 live in at most n iterations. However, the structure of S^0 does depend on the numbers assigned to the cells of the network, and hence all possible numbering schemes should be considered. In fact, a condition that is independent of any particular numbering of the cells may only be obtained by an analysis of the graph underlying the initial state, rather than its matrix representation. We pursue this approach by first defining the initial state graph.

Definition 8: The initial state graph of a network N is a labeled directed graph $G_0 = (V, E, L)$, where V is a set of nodes corresponding to the cells in N, E is a set of edges corresponding to the links in N, and $L : E \longrightarrow \{IRE, IWE\}$ is a function that assigns a label to each edge in E.

Definition 9: If e and v are two nodes in V, then v is w-reachable from e, denoted by $e \leadsto v$, if G_0 contains a path P from e to v, such that every edge in P is IWE. Also, v is an R-neighbor of e, denoted by $e \rightarrow v$, if G_0 contains an IRE edge directed from e to v.



(a) (b) (c)

Figure 1 - Initial configurations leading to deadlock.

Particular configurations in G_0 may be easily shown to cause deadlock. Consider, for example, the following cases:

Case 1: If there exist a loop with IWE edges (Fig 1a).
Case 2: If there exist an IRE edge that bypasses one or more IWE edges (Fig 1b).
Case 3: If there exist two IRE cross links between two IRE paths (Fig 1c).

Clearly, the existence of any of the above cases is sufficient to cause deadlock. However, is their absence sufficient to ensure deadlock free operation?. The positive answer to this question is given by the following theorem.

Theorem 3: An initial state of a self-timed network N is live if and only if its initial state graph $G_0 = (V,E,L)$ satisfies all of the following conditions:

C1) There does not exist a node $e \, \varepsilon \, V$ with $e \rightsquigarrow e$,

C2) There does not exist two nodes $e, v \, \varepsilon \, V$ with $v \rightsquigarrow e$ and $v \rightarrow e$,

C3) For any $k \geq 1$, there does not exist $2k$ nodes $e_i, v_i \, \varepsilon \, V$, $i = 0, \cdots, k-1$ with $v_{k-1} \rightarrow e_0$, $v_{i-1} \rightarrow e_i$, $i = 1, \cdots, k-1$, and $v_i \rightsquigarrow e_i$, $i = 0, \cdots, k-1$. []

Proof: Let $G_0 = (V,E,L)$ be an initial state graph of N, and let S^0 be the initial state matrix corresponding to any numbering of the cells of N.

First assume that **C1** is violated. Then there exists a cycle $e_1, \cdots, e_m$ such that, for $i = 1, \cdots, m-1$, $S_{e_i, e_{i+1}} = -1$, and $S_{e_m, e_1} = -1$. By (4c) and (4a) of Lemma 3, this gives

$$E^{(1)}_{w,e_1} < E^{(1)}_{r,e_2} < E^{(1)}_{w,e_2} < E^{(1)}_{r,e_3} < \cdots < E^{(1)}_{w,e_m} < E^{(1)}_{r,e_1}$$

But from (4a), $E^{(1)}_{r,e_1} < E^{(1)}_{w,e_1}$. Hence, neither $E^{(1)}_{w,e_1}$ nor $E^{(1)}_{r,e_1}$ do occur in any S^0-feasible sequence. The same technique used in the proof of Theorem 2 may be used here, with $\alpha = \{e_1\}$ to prove that the initial state is not live. Similarly, we may show that the initial state is not live if any of **C2** or **C3** is violated. That is, the initial state is live only if C1, C2 and C3 are satisfied.

In order to prove the "if" part of the theorem, we assume that algorithm LIVE is applied to S^0, and that during its execution, we keep a partitioning of the nodes in G_0 into three disjoint sets, namely T_o, T_r and T_w, as follows:

[] Initially, let $T_o = V$ and $T_r = T_w = \phi$.
[] Whenever an instance $E^{(1)}_{r,i}$ is simulated in LIVE, move node i from T_o to T_r.
[] Whenever an instance $E^{(1)}_{w,i}$ is simulated in LIVE, move node i from T_r to T_w.

Upon termination of LIVE, if S^0 is declared live, then $T_w = V$. On the other hand, if S^0 is declared non-live, then $T_r \cup T_o \neq \phi$, and the following holds:

B1) For any node $e \ \varepsilon \ T_o$, there exists at least one IWE edge directed to e from a node $v \ \varepsilon \ T_o \cup T_r$.

B2) For any node $\overline{v} \ \varepsilon \ T_r$, there exists at least one node $\overline{e} \ \varepsilon \ T_o$ such that $\overline{v} \rightarrow \overline{e}$.

B3) $T_o \neq \phi$.

Following is the proof of B1, B2 and B3: When LIVE declares S^0 non live, then
1) for any $e \ \varepsilon \ T_o$, there exists some $v \ \varepsilon \ V$ such that $S[v,e] = -1$. But $v \ \varepsilon \ T_w$ implies that $S[v,e] = 1$. Hence, $v \ \varepsilon \ T_o \cup T_r$. Clearly, neither $E^{(1)}_{r,e}$ nor $E^{(1)}_{w,v}$ did execute. That is $S^m_{v,e} = S[v,e] = -1$, which proves that $l_{v,e}$ is IWE.
2) for any $\overline{e} \ \varepsilon \ T_r$, there exists some $\overline{e} \ \varepsilon \ V$ such that $S[\overline{v},\overline{e}] = 1$. But $\overline{e} \ \not\varepsilon \ T_r$ implies that $S[\overline{v},\overline{e}] = -1$. Hence, $\overline{e} \ \varepsilon \ T_r$. As in 1) above, $S^m_{\overline{v},\overline{e}} = S[\overline{v},\overline{e}] = 1$, which proves that $l_{\overline{v},\overline{e}}$ is IRE.
3) if $T_o = \phi$, then $T_r \neq \phi$. From B2, this implies the existence of some node $\overline{e} \ \varepsilon \ T_o$. Hence, $T_o \neq \phi$.

Now, assuming that C1, C2 and C3 are satisfied in G_0, we prove, by contradiction, that there cannot exists two disjoint subsets $T_o, T_r \subseteq V$, $T_o \cup T_r \neq \phi$, such that B1, B2 and B3 are satisfied. Assume that such T_o and T_r do exists, and construct two subsets $A_0 \subseteq T_o$ and $A_r \subseteq T_r$ as follows:

1) Start with $A_o = \{e_0\}$ and $A_r = \{v_0\}$, such that $v_0 \ \varepsilon \ T_r$ and $v_0 \rightsquigarrow e_0$.

2) For $k = 1, 2, \cdots$

   2.1) Chose $e_k \ \varepsilon \ T_o$ such that $v_{k-1} \rightarrow e_k$ (e_k exists by B2). If $e_k \ \varepsilon \ A_o$ then stop the construction, else set $A_o = A_o \cup \{e_k\}$.

   2.2) Chose $v_k \ \varepsilon \ T_r$ such that $v_k \rightsquigarrow e_k$ (the existence of v_k is discussed below). If $v_k \ \varepsilon \ A_r$ then stop the construction, else set $A_r = A_r \cup \{v_k\}$.

In order to prove the existence of v_0 and v_k in steps 1 and 2.2, we consider any node $x_0 \ \varepsilon \ T_o$ and prove that x_0 is W-reashable from some node $y_0 \ \varepsilon \ T_r$. By B1, there exists $x_1 \ \varepsilon \ T_o \cup T_r$ such that $x_1 \rightsquigarrow x_0$. If $x_1 \ \varepsilon \ T_r$, then take $y_0 = x_1$, else x_1 should be in T_o. Again, by B1, there exists $x_2 \ \varepsilon \ T_o \cup T_r$ such that $x_2 \rightsquigarrow x_1$. If $x_2 \ \varepsilon \ T_r$ then take $y_0 = x_2$, else x_2 should be in T_0, and by C1 $x_2 \neq x_i$, $0 \leq i \leq 2$. This argument can be repeated at most d times, where d is the number of nodes in T_o, before a node $y_0 \ \varepsilon \ T_r$ with $y_0 \rightsquigarrow x_0$ is found.

Given that T_0 and T_r are finite, then the construction of A_o and A_r should terminate. If it terminates in step 2.1, then $e_k = e_j$ for some $j \leq k-1$. If $j = k-1$ then $v_{k-1} \rightsquigarrow e_{k-1}$ and $v_{k-1} \rightarrow e_{k-1}$, which contradicts C2. If $j < k-1$, then $v_{k-1} \rightarrow e_j$, $v_{i-1} \rightarrow e_i$, $i = j+1, \cdots, k-1$ and $v_i \rightsquigarrow e_i$, $i = j, \cdots, k-1$. This clearly contradicts C3.

A similar argument may be given if the construction terminates at step 2.2. Hence, if C1, C2 and C3 are satisfied in G_0, then a partitioning of G_o that satisfies B1, B2 and B3 cannot exist .[]

The first two conditions are formal expressions for cases 1 and 2 shown in Fig 1. The third condition is a general form of case 3, where k IRE cross links between k IWE paths form a cycle.

6. CONCLUSION

We considered a class of self-timed networks where each cell does not read its input unless all inputs are available and does not write its outputs until all previous results are consumed. For this specified class of networks , we proved that the liveness of a computation depends uniquely on the initial state of the network. Moreover, we proved that a network is deadlock-free if and only if each cell completes the execution of at least one read/write cycle. This means that only the initial segment of any simulation may provide a proof of liveness. We also derived sufficient and necessary conditions for the verification of liveness without any simulation.

References

1. S. Brookes, C. Hoare, and A. Roscoe, ''A Theory of Communicating Sequential Processes,'' *J. of ACM*, vol. 31,3, 1984.
2. P. Caspi and N. Halbwachs, ''Algebra of Events: A model for Parallel and Real Time Systems,'' *Proc. of the 1982 International Conf. on Parallel Preocessing*, pp. 150-159.
3. K. Chandy and J. Misra, ''Deadlock Absence Proofs for Networks of Communicating Processes,'' *Information Processing Letters*, vol. 9,4, pp. 185-189, 1979.
4. H. T. Kung, ''Why Systolic Architecture,'' *Computer*, vol. 15, pp. 37-46, 1982.
5. L. Lamport, ''Time, Clocks and Ordering of Events in a Distributed System,'' *CACM* , vol. 21,7, pp. 558-565, 1978.
6. M. McFarland and C. Parker, ''An Abstract Model of Behavior for Hardware Descriptions,'' *IEEE Trans. on Computers*, vol. 32,7, pp. 621-637, 1983.
7. R. G. Melhem, ''Verification of a Class of Deadlock-free, Self-timed Computational Arrays,'' Technical report ICMA-84-76, 1984. The University of Pittsburgh
8. R. G. Melhem and W. C. Rheinboldt, ''A Mathematical Model for the Verification of Systolic Networks,'' *SIAM J. on Computing*., vol. 13, no. 3, pp. 541-565, Aug. 1984.
9. R. G. Melhem, ''A Language for the Simulation of Systolic Architectures,'' *Proc. of the 12th International Symposium on Computer Architecture - Boston - Massachusetts*, June 1985.
10. B. Mishra and E. Clarke, ''Automatic and Hierarchical Verification of Asynchronous Circuits Using Temporal Logic,'' *Tec. Report CMU-CS-83, dept. of Computer Science, Carnegie-Mellon University*, 1983.
11. C. Seitz, ''System Timing,'' in *Introduction to VLSI Systems*, ed. C. Mead and L. Conway, pp. 218-262, 1980.

ON SUPERVISOR REDUCTION IN DISCRETE EVENT SYSTEMS

A.F. Vaz and W.M. Wonham
Systems Control Group
Dept. of Electrical Engineering
University of Toronto
Toronto, Ontario
CANADA M5S 1A4

ABSTRACT

A supervisory controller (supervisor) **S** for a discrete event system can be modelled on a recognizer **R** for the language corresponding to the supervisory task to be accomplished. It is shown that simpler "reduced" supervisors can be constructed by the use of covers of the state set of **R**; and that any "normal" supervisor is a reduction of **R** in this sense. The reduction procedure is time-exponential with respect to the size of the state set of **R**.

1. Introduction

Discrete event systems (DES) can be described as systems or processes that are discrete, asynchronous and possibly nondeterministic. Examples of systems of this type include manufacturing systems, communication protocols and the like. A control theory for a general class of DES has been proposed in [RW83]; see also the summary articles [RW84], [WR84a,b], [W85b]; and the detailed reports [WR83a,b], [W85], [LW85]. The reader is referred to these sources for general background and motivation, as well as for details of the mathematical setup, on which we shall draw freely. Proofs and technicalities omitted in this conference paper for lack of space are provided in the report [VW85].

In our theory, the DES to be controlled is modelled as a tuple

$$\mathbf{G} = (\Sigma, \Sigma_u, \Sigma_c, Q, \delta, q_0, Q_m)$$

where

$$\Sigma = \Sigma_u \cup \Sigma_c$$

is an alphabet of event labels consisting of uncontrollable elements (Σ_u) and controllable elements (Σ_c); Q is the state set; $\delta : \Sigma \times Q \rightarrow Q$ is the state transition function (a partial function, defined for each $q \in Q$ on a subset of the $\sigma \in \Sigma$); $q_0 \in Q$ is the initial state; and $Q_m \subset Q$ is the subset of "marked" states. Let Σ^* denote the set of all finite strings s of elements of Σ, including the empty string 1. By finite iteration of δ starting from q_0, **G** is established as the generator of a formal language

$$L(\mathbf{G}) := \{s \in \Sigma^* \mid \delta(s, q_0) \text{ is defined}\}$$

while

$$L_m(\mathbf{G}) := \{s \in \Sigma^* \mid \delta(s, q_0) \in Q_m\}$$

is the subset of $L(\mathbf{G})$ of "marked" strings representing "completed tasks" of the DES that **G** is intended to model.

A *supervisor* for **G** is a pair $\mathbf{S} = (S, \psi)$, where

$$S = (\Sigma, X, \xi, x_0, X_m)$$

is an automaton with state set X, initial state x_0, a marked subset $X_m \subset X$, and (partial) transition function $\xi : \Sigma \times X \to X$; while

$$\psi : \Sigma \times X \to \{0, 1, dc\}$$

is a *control law* such that

$$\psi(\sigma, x) = 1, \; x \in X, \; \sigma \in \Sigma_u$$

$$\psi(\sigma, x) \in \{0, 1, dc\}, \; x \in X, \; \sigma \in \Sigma_c$$

S is considered to be driven externally by the stream of output elements σ of **G**; while in turn, with S in state x, the transitions (labelled) σ of **G** are subject to the control $\psi(\sigma, x)$. If $\psi(\sigma, x)=0$, then σ is "disabled" (prohibited from occurring), while if $\psi(\sigma, x)=1$ then σ is "enabled" (permitted but not forced to occur). The notation dc ("don't care") means that assignment to (σ, x) of either 0 or 1 would make no difference to the behavior of **G**. The transition function $\delta : \Sigma \times Q \to Q$ of **G** is then replaced by $\delta_\psi : \Sigma \times X \times Q \to Q$, where

$$\delta_\psi(\sigma, x, q) = \delta(\sigma, q)$$

if $\psi(\sigma, x)=1$ and is undefined otherwise. In this way one obtains a closed-loop feedback structure **S/G** ("**G** supervised by **S**"). **S/G** is thereby another DES

$$\mathbf{S/G} = (\Sigma, \Sigma_u, \Sigma_c, X \times Q, \xi \times \delta_\psi, (x_0, q_0), X_m \times Q_m)$$

with (partial) transition function defined by

$$(\xi \times \delta_\psi)(\sigma, (x, q)) := (\xi(\sigma, x), \delta_\psi(\sigma, x, q))$$

whenever both components are defined on the right. We denote the corresponding closed-loop languages by $L(\mathbf{S/G})$, $L_m(\mathbf{S/G})$. For simplicity we temporarily confine attention to $L(\mathbf{S/G})$.

It was shown in [RW83] that, if $K \subset \Sigma^*$ is a language such that there exists a supervisor $\mathbf{S} = (S, \psi)$ with

$$K = L(\mathbf{S/G})$$

then S can be modelled on a recognizer for K. In practice the transition structure of K is often considerably more complex than is necessary for S to perform its role in **S**. It is then of interest to reduce **S** to a simpler supervisor, say $\bar{\mathbf{S}}$, such that control action is preserved, namely

$$K = L(\bar{\mathbf{S}}/\mathbf{G}).$$

In [RW83] such reduction was achieved by a suitable partition of the state set X of S. The objectives of the present article are 1. to generalize this construction by the use of "covers" instead of partitions; 2. to obtain a generalized version of the Quotient Structure Theorem of [RW83]; and 3. to present an algorithm by which a minimal (i.e. "optimal") state structure for $\bar{\mathbf{S}}$ can be computed.

2. Main Structural Results

To state the main results we define a *cover* of $\mathbf{S}$ to be a family $\mathbf{C} = \{X_i, i \in I\}$ of subsets of X with the following properties (cf. [Z68]). Here the notation $\xi(\sigma, x)!$ means $\xi(\sigma, x)$ is defined.

$$(\forall i) X_i \neq \emptyset;$$

for a subset $I_m \subset I$,

$$X_m = \bigcup \{X_i \mid i \in I_m\}, \quad X - X_m = \bigcup \{X_i \mid i \in I - I_m\};$$

$$(\forall i, j) X_i \subset X_j \Rightarrow i = j;$$

$$(\forall i, \sigma).(\exists y \in X_i) \xi(\sigma, y)!$$

$$\Rightarrow (\exists j)(\forall x \in X_i) \xi(\sigma, x)! \Rightarrow \xi(\sigma, x) \in X_j$$

{for brevity we write this property as

$$(\forall i, \sigma)(\exists j) \xi(\sigma, X_i) \subset X_j\};$$

$$(\forall i, \sigma)(\forall x, y \in X_i) \psi(\sigma, x) \neq dc \neq \psi(\sigma, y)$$

$$\Rightarrow \psi(\sigma, x) = \psi(\sigma, y).$$

Thus a cover of $\mathbf{S} = (S, \psi)$ is simply a covering of the state X of S by nonempty subsets, such that the marked states (X_m) are covered separately from the unmarked states $(X - X_m)$, no cover element contains a proper subset that is a cover element, the cover elements behave consistently under the action of the transition function ξ, and a cover element exhibits uniform control action at those states where control matters.

The case of interest is $|\mathbf{C}| < |X|$, where $|\cdot|$ denotes the number of elements (assumed finite) of its argument. Given $\mathbf{C}$ with this property, we define a *reduced* supervisor $\bar{\mathbf{S}} = (\bar{S}, \bar{\psi})$ as follows.

$$\bar{S} = (\Sigma, \bar{X}, \bar{\xi}, \bar{x}_0, \bar{X}_m)$$

$$\bar{X} = I$$

$$\bar{X}_m = I_m$$

and $\bar{\xi}, \bar{x}_0, \bar{\psi}$ are defined (perhaps not uniquely) in the natural way consistent with the properties of $\mathbf{C}$. In general the reduced supervisor is determined by selections that in part are arbitrary, but this fact is of no account in what follows. Our first main result is

Theorem 1 (Supervisor reduction)

$$L(\bar{\mathbf{S}}/\mathbf{G}) = L(\mathbf{S}/\mathbf{G}), \quad L_m(\bar{\mathbf{S}}/\mathbf{G}) = L(\mathbf{S}/\mathbf{G})$$

For our second main result we assume that $\mathbf{S}$ is an arbitrary supervisor for $\mathbf{G}$ and that (for simplicity) $X_m = X$. Denote $K := L(\mathbf{S}/\mathbf{G})$. Let $\hat{S} = (\Sigma, \hat{X}, \hat{\xi}, \hat{x}_0, \hat{X})$ be a recognizer for K. Assume finally that $\mathbf{S}$ satisfies the mild technical condition that it is "normal", i.e. *standard*, *nonredundant*, and *weakly K - reduced* [VW85].

Theorem 2 (Generalized quotient structure)

There exists a control law

$$\hat{\psi} : \Sigma \times \hat{X} \to \{0,1,dc\}$$

such that $\hat{\mathbf{S}} = (\hat{S}, \hat{\psi})$ is a supervisor for $\mathbf{G}$ with the property

$$L(\hat{\mathbf{S}}/\mathbf{G}) = L(\mathbf{S}/\mathbf{G}), \quad L_m(\hat{\mathbf{S}}/\mathbf{G}) = L_m(\mathbf{S}/\mathbf{G})$$

Furthermore, with **S** normal, there is a cover **C** of $\hat{\mathbf{S}}$ such that the corresponding reduced version of $\hat{\mathbf{S}}$ can be identified with **S**.

Here the identification is accomplished by an isomorphism, defined in a natural way. This result captures the intuition that any "reasonable" (i.e. normal) supervisor **S** can be derived from (or "is a high-level version of") a supervisor $\hat{\mathbf{S}}$ modelled on the grammatical structure of the supervisory task K to be accomplished.

3. An Algorithm for Optimal Reduction

From Theorem 1 we know that the synthesis of a minimal supervisor entails two steps: the search for a cover with a minimal number of elements, and the construction of a supervisor from that cover. We need the following definitions. Two states $x, y \in X$ are *control-compatible* if

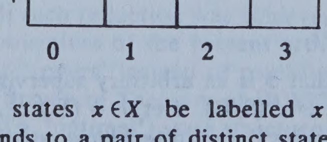
$$(\forall \sigma \in \Sigma) \psi(\sigma, x) \neq dc \neq \psi(\sigma, y) \Rightarrow \psi(\sigma, x) = \psi(\sigma, y)$$

The (one-step) *transition pairs* of a state pair $(x, y) \in X \times X$ is the set of pairs $TP(x, y)$ defined by

$$TP(x,y) := \{(x',y')x',y' \in X\,;\ x' \neq y';\ \&\ (\exists \sigma \in \Sigma)$$
$$\xi(\sigma, x) = x' \ \ \&\ \ \xi(\sigma, y) = y'\}$$

Two states $x, y \in X$ are *compatible* if (i) x, y are control-compatible, (ii) $x \in X_m$ iff $y \in X_m$, and (iii) all state pairs $(x', y') \in TP(x, y)$ satisfy (i) and (ii). A nonempty set of states $B \subset X$ is a *block* if every state pair (x, y) with $x, y \in B$ is compatible.

Compatible state pairs are determined by use of a *merger table* (cf. [PU59] and [K78]). A table for 5 states is shown below.



Let the states $x \in X$ be labelled $x_0, x_1, \cdots$. Each cell of the merger table corresponds to a pair of distinct states (x_i, x_j). For $i > j$ let cell (i, j) denote the cell in row i, column j of the table. The cells are filled in according to the following procedure.

```
procedure fillin;  {fills in merger table}
  begin
```

```
for each pair $(x_i,x_j) \in X \times X$ with $i > j$ do
  begin
    compatible:= true;
    if (($x_i$ and $x_j$ are not control-compatible)
       or ($x_i \in X_m$ and $x_j \in X - X_m$) or ($x_i \in X - X_m$ and $x_j \in X_m$))
    then compatible:= false;
    if compatible = false then cell $(i,j)$:= false;
    if compatible = true then
       begin
         form $TP(x_i,x_j)$;
         if $TP(x_i,x_j) = \varnothing$ then cell $(i,j)$:= true
           else cell $(i,j)$:=$TP(x_i,x_j)$
       end
  end
end.
```

When the merger table is completed, cells containing transition pairs are assigned an appropriate boolean value by the resolution procedure described in [PU59] and [K78]. It is then the case that a state pair (x_i,x_j) is compatible (i.e. is a block) iff cell (i,j) = true. Blocks containing more than two states are constructed in a similar way. All possible blocks must be listed. For an n-state supervisor there are at worst about 2^n blocks. The block list is searched for all singletons, pairs, triples etc. of blocks until a minimal cover is found. The computation is exponential in time and so is feasible only for small values of n.

From a given supervisor $\mathbf{S} = (S,\psi)$ with $S = (\Sigma, X, \xi, x_0, X_m)$ we construct a reduced supervisor $\overline{\mathbf{S}} = (\overline{S},\overline{\psi})$ with $\overline{S} = (\Sigma, Y, \eta, y_0, Y_m)$. Let $\mathbf{C} = \{B_0, B_1, \ldots, B_r\}$ be a minimal cover with $x_0 \in B_0$. The synthesis is carried out by the following procedure.

```
procedure synthesize;
  {constructs a reduced supervisor from given cover}
  begin
    {construct transition function $\eta(\sigma,y)$}
      for $i$ := 0 to $r$ let $y_i$ denote $B_i$;
      for $i$ := 0 to $r$ do
        begin
          for each $\sigma \in \Sigma$ do
            begin
              $R := \{x' : x' = \xi(\sigma,x), x \in B_i\}$;
              if $R \neq \varnothing$ then
                begin
                  select $k$ such that $R \subset B_k$;
                  $\eta(\sigma,y_i) := y_k$
                end
            end
        end
    {construct control $\overline{\psi}(\sigma,y)$}
    for $i$ := 0 to $r$ do
      for each $\sigma \in \Sigma$ do
```

```
        if $\exists x \in B_i$ such that $\psi(\sigma,x) \neq dc$
            then $\bar{\psi}(\sigma,y_i) := \psi(\sigma,x)$
            else $\bar{\psi}(\sigma,y_i) := dc$
    end.
```

A program based on these procedures has been tested on the VAX 11/780. For the 9-state supervisor of Example 1 in [RW83] a reduction to 4 states was achieved. The computation generated 37 blocks and required 11.4 secs. of CPU time. For the 6-state supervisor of Example 1 in [LW85] a reduction to 4 states was achieved in 5.6 secs. of CPU time.

4. Conclusions

The main structural results provide a generalization of the supervisor reduction scheme and quotient structure theorem of [RW83]. Examples show that reduction by covers may indeed lead to supervisors with smaller state sets than those achievable by partitions. From a theoretical viewpoint, the generalized quotient structure theorem improves on the result of [RW83] in placing a less stringent requirement (normal versus "$L(S/G)$-reduced") on the class of supervisors that are shown to be "quotients". The fact that covers may improve on partitions depends essentially on the feature that the supervisor transition function and control law are generally partial functions; if both of these are total functions, any reduction by a cover can be shown to be equivalent to a reduction by a suitable partition with at most as many elements. In practice, of course, the exponential complexity of the computation of a minimal cover will rule out covers as a design approach in all but relatively simple cases. Nevertheless, in combination with other techniques of supervisor synthesis (notably decentralization and modularity) reduction by covers can play a useful role.

5. References

[K78] Z. Kohavi. Switching and Finite Automata Theory. McGraw-Hill, New York, 1978.

[LW85] F. Lin, W.M. Wonham. On the computation of supremal controllable sublanguages. Systems Control Group Rpt. #8506, Dept. of Electl. Engrg., Univ. of Toronto, April 1985.

[PU59] M.C. Paull, S.H. Unger. Minimizing the number of states in incompletely specified sequential switching functions. IRE Trans. on Electronic Computers **8**, 1959, pp. 356-367.

[RW83] P.J. Ramadge, W.M. Wonham. Supervisory control of a class of discrete event processes. Systems Control Group Rpt. #8311, Dept. of Electl. Engrg., Univ. of Toronto, October 1983; revised November 1984.

[RW84] P.J. Ramadge, W.M. Wonham. Supervisory control of a class of discrete event processes. Proc. Sixth Intnl. Conference on Analysis and Optimization of Systems, Nice, June 1984. In A. Bensoussan, J.L. Lions (Eds.), Analysis and Optimization of Systems, Lecture Notes on Control and Information Sciences No. 63, Springer-Verlag, Berlin, 1984; Part 2, pp.

447-498.

[VW85] A.F. Vaz, W.M. Wonham. On supervisor reduction in discrete event systems. Systems Control Group Rpt. #8513, Dept. of Electl. Engrg., Univ. of Toronto, October 1985.

[WR83a] W.M. Wonham, P.J. Ramadge. On the supremal controllable sublanguage of a given language. Systems Control Group Rpt. #8312, Dept. of Electl. Engrg., Univ. of Toronto, November 1983; revised November 1984.

[WR83b] W.M. Wonham, P.J. Ramadge. On modular synthesis of supervisory controls for discrete event processes. Systems Control Group Rpt. #8314, Dept. of Electl. Engrg., Univ. of Toronto, December 1983; revised January 1985.

[WR84a] W.M. Wonham, P.J. Ramadge. On the supremal controllable sublanguage of a given language. Proc. 23rd IEEE Conf. on Decision and Control, IEEE Control Systems Society, New York, December 1984, pp. 1073-1080.

[WR84b] W.M. Wonham, P.J. Ramadge. On modular synthesis of supervisory controls for discrete event processes. Proc. Intnl. Conf. on Computers, Systems and Signal Processing, IEEE and I.I.Sc., Bangalore, December 1984, pp. 500-504.

[W85a] W.M. Wonham. On control of discrete event systems. Systems Control Group Rpt. #8508, Dept. of Electl. Engrg., Univ. of Toronto, July 1985.

[W85b] W.M. Wonham. On control of discrete event systems. Seventh Intnl. Symp. on the Mathematical Theory of Networks and Systems (MTNS-85), Stockholm, June 1985. [Proceedings to be published.]

[Z68] H.P. Zeiger. Cascade decomposition of automata using covers. In M.A. Arbib (Ed.), Algebraic Theory of Machines, Languages, and Semigroups. Academic Press, New York, 1968; pp. 55-80.

MODIFIED PETRI NETS FOR HIERARCHICAL SIMULATION AND CONTROL OF MANUFACTURING SYSTEMS[1]

BRUCE H. KROGH and CAROLYN L. BECK
Department of Electrical and Computer Engineering
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

A methodology is presented for constructing models of manufacturing processes for simulation and discrete controller design. The models represent the discrete event evolution of the system as well as features of the underlying continuous processes. The structure of the discrete-level control is modeled with *modified Petri nets* which are synthesized from single resource activity cycles. Construction of the net constitutes a design of the discrete control logic with guaranteed properties based on extensions of previous results in Petri net theory. The model includes error recovery loops and other real-time decision structures.

SUMMARY

Design of manufacturing systems is generally approached on two distinct levels. On one level the system is viewed as the interaction of discrete, asynchronous, concurrent events which must be coordinated to enforce precedence relations and allocate resources. On a lower level the actual operations performed by the manufacturing system are viewed as continuous-parameter processes which evolve in continuous time. This paper presents an approach to the modeling of manufacturing systems which incorporates aspects of both levels of the process.

For a given system the basic elements of the model are defined by the following steps:

1. Decompose the process into *operations* (e.g., moves, insertions, fastenings, etc.)
2. List the *resources* required for each operation (e.g., tools, robots, parts, fixtures, etc.)
3. Determine the *resource attributes* affected by, or involved in, each operation.
4. Construct the process model for each operation, i.e., choose representations for the *tolerance requirements*, the *operation transformation*, and the *operation delay.*
5. List the discrete resource states which are preconditions and postconditions of all operations, including failure conditions if error recovery is desired.
6. Define the mapping of resource attributes into discrete resource states.

To model the discrete decision and control structure of the manufacturing system we use a modification of Petri Nets (PNs) which were originally defined to model concurrency in computer systems [1]. A Petri Net (PN) is a four-tuple, (P,T,IN,OUT), where $P=\{p_1, \ldots, p_n\}$ is the set of n *places,* $T=\{t_1, \ldots, t_m\}$ is the set of m *transitions,* and IN (OUT) is an mxn binary matrix indicating the places mapped into (out of) a transition. C = IN - OUT is the PN *incidence matrix.* A *marking* $\mathbf{m}$ of a PN is an integer-valued row-vector indicating then number of *tokens* in each place. A transition is *enabled* when at least one token resides in each of its input places, and an enabled transition *fires* by removing one token from each of its input places and placing one token on each of its output places. Given an initial marking $\mathbf{m}^o$, the set of *reachable markings* $R(\mathbf{m}^o)$ is defined as the set of markings which can be generated by a valid firing sequence starting with marking $\mathbf{m}^o$.

Properties of a Petri net, such as *liveness* and *safeness* [1], are related to the net *invariants.* A binary n-vector $\mathbf{y}$ is a *p-invariant* if $C\mathbf{y}=0$, in which case $\mathbf{m}\mathbf{y} \equiv \mathbf{m}^o\mathbf{y}$ for all $\mathbf{m}$ in $R(\mathbf{m}^o)$. Place p_j is *in* a p-invariant $\mathbf{y}$ if $y_j=1$. The following propositions [2] concerning invariants of combined nets are extensions of previous results in PN theory [3], [4], [5].

---

[1] This research was supported in part by the Xerox Corporation, the IBM Corporation, Ben Franklin Partnership Program (Commonwealth of Pennsylvania), and that National Science Foundation under research grant DMC-8451493.

Proposition 1. If Petri nets PN_1, PN_2 with p-invariants $\mathbf{y}^1, \mathbf{y}^2$, respectively, are combined along a common chain which is terminated on both ends by *places,* then the union of the sets of places in $\mathbf{y}^1$ and $\mathbf{y}^2$ corresponds to a p-invariant for the resulting net.

Proposition 2. If Petri nets PN_1, PN_2 with p-invariants $\mathbf{y}^1, \mathbf{y}^2$, respectively, are combined along a common chain which is terminated on both ends by *transitions*, then the set of places in $\mathbf{y}^1$ and the set of places in $\mathbf{y}^2$ each correspond to a p-invariant for the resulting net.

The application of PNs to flexible manufacturing systems was proposed by Dubois and Stecke [6] for *decision-free* systems. To model decision-based (*controlled*) manufacturing systems, we define a class of Modified Petri Nets (MPNs) in which there are two types of places, *operation places* (*o-places*) and *resource state places* (*rs-places*). Along any *chain* in a valid MPN graph, the rs-places and o-places alternate. Thus, the *transitions* in an MPN can be decomposed into two classes: *input transitions,* which immediately precede o-places; and *output transitions* which immediately follow o-places. Each transition is connected to a *single* o-place. In an MPN model of a manufacturing system, the discrete resource states are mapped into rs-places and operations are mapped into o-places.

To apply the results for combining PNs to the MPN structures, it is necessary to define the common chains so that the resulting nets are still valid MPNs. We restrict the common chains to the following three classes. Type 1: Chains starting with an input transition and ending with an output transition; Type 2: chains terminated on both ends by o-places; Type 3: chains terminated on both ends by rs-places.

The synthesis rules are summarized as follows:

1. Construct all simple resource activity cycles.
2. For "AND" discrete resource state conditions for operations, combine MPNs with common Type 1 chains.
3. For "OR" discrete resource state conditions for operations, combine MPNs with common Type 2 chains.
4. For *shared* resources, combine MPNs with common Type 3 chains.

Based on the propositions above, construction of the MPN following these systhesis rules guarantees the complete net is live and safe with respect to a class of initial markings. Thus, the resulting control logic is free of deadlocks and inconsistencies. The graphical representation of MPNs and application of these results to the simulation and control of a two-arm robotic assembly cell are described in reference [2].

References

1. J.L. Peterson, *Petri Net Theory and the Modeling of Systems,* Prentice-Hall, Inc., Englewood Clifss, NJ, 1981.

2. C.L. Beck, "Modeling and Simulation of Flexible Control Structures for Automated Assembly Systems", Master's thesis, Carnegie-Mellon University, 1985.

3. T. Agerwala, "Some Applications of Petri Nets", *Proc. '78 National Electronics Conf.*, Vol. 23, October 1978, pp. 149-154.

4. Y. Narahari and N. Viswanadham, "A Petri Net Approach to the Modelling and Analysis of Flexible Manufacturing Systems", *J. Annals of Operations Research,* to appear 1985.

5. K. Lautenbauch and H. Schmid, "Use of Petri Nets for Proving Correctness of Concurrent Process Systems", *IFIP,* 1974, pp. 187-191.

6. D. Dubois and K.E. Stecke, "Using Petri Nets to Represent Production Processes", *Proceedings of the 22nd IEEE CDC,* Dec 1983, pp. 1062-1067.

ON THE COMPUTATION OF SUPREMAL CONTROLLABLE SUBLANGUAGES

F. Lin and W.M. Wonham
Systems Control Group
Dept. of Electrical Engineering
University of Toronto
Toronto, Ontario
CANADA M5S 1A4

ABSTRACT

In this report an algorithm is developed to compute the supremal controllable sublanguage of a given language in case the languages involved are regular. To implement the algorithm, a computer program has been written in Pascal. Two examples are given, of which one illustrates an application to computer communication networks.

1. Introduction

In [1], a theory was developed for the supervisory control of a class of discrete event processes. It was shown that the concept of *controllable language* plays a basic role in the existence theory of supervisory controls. In order to synthesize a supervisory control, the *supremal controllable sublanguage* of a given "target" language L must be computed.

In [2], the supremal controllable sublanguage S of a given language L was characterized as the largest fixpoint of a certain operator Ω. In the case where the languages involved are regular, it was shown that the fixpoint S can be computed as the limit of the (finite) sequence $\{K_j\}$ given by

$$K_0 = L$$

$$K_{j+1} = \Omega(K_j), \quad j = 0,1,...$$

In this report an algorithm to compute the supremal controllable sublanguage is developed. To implement the algorithm a computer program SUPCON was written in Pascal. Two examples are given, of which one illustrates an application to computer communication networks.

We first recall some definitions from [2].

Let Σ be a (nonempty) finite *alphabet* of elements σ, and as usual denote by Σ^* the set of all finite strings of elements of Σ, including the empty string 1.

A subset $L \subset \Sigma^*$ is a *language* over Σ. A string $s \in L$ is a *word* of L.

If $s, s', t \in \Sigma^*$ with $s't = s$, then s' is a *prefix* of s. The *closure* of L is the language $\bar{L}$ consisting of all the prefixes of words in L. L is *closed* if $L = \bar{L}$.

Let $L \subset \Sigma^*$. Strings $s, t \in \Sigma^*$ are *equivalent* (mod L), written

$$s \equiv t \,(mod\ L) \quad \text{or} \quad s \equiv_L t$$

if

$$\{s' : s' \in \Sigma^* \ \& \ ss' \in L\} = \{t' : t' \in \Sigma^* \ \& \ tt' \in L\}$$

In other words, two strings are equivalent (mod L) if they are each "continuable" in the same ways to form completed words of L. We shall write

$$\|L\| := \mathrm{card}(\Sigma^* / \equiv_L),$$

namely $\|L\|$ is the (possibly denumerable) number of equivalence classes of $\equiv_L$ in Σ^*.

The language L is *regular* if $\|L\| < \infty$.

Let M be a fixed language over Σ and let Σ_0 be a fixed subset of Σ. A language $K \subset \Sigma^*$ is *controllable* (with respect to M and Σ_0) if

$$\bar{K}\Sigma_0 \cap M \subset \bar{K}$$

Let $L \subset \Sigma^*$ be arbitrary. Then the *supremal controllable sublanguage* of L is the language

$$\sup\underline{C}(L) := U\{K : K \subset L \text{ and } K \text{ is controllable}\}$$

It was shown in [1] that $\sup\underline{C}(L)$ is well defined (although it may be empty), is controllable, and contains every controllable sublanguage of L.

Let $\mathbf{L}$ be the set of all languages over Σ (i.e. $\mathbf{L}$ = power set of Σ^*) and define the operator

$$\Omega : \mathbf{L} \to \mathbf{L}$$

according to

$$\Omega(K) = L \cap \sup\{T : T \subset \Sigma^*,\ T = \bar{T}\ \ \&\ \ T\Sigma_0 \cap M \subset \bar{K}\}$$

It is easy to check (cf. the proof of Proposition 7.1 in [1]) that the condition in { } is closed under arbitrary unions, and so Ω is well defined.

The following result was obtained in [2].

THEOREM

In case the languages L and M are regular, the sequence of languages K_j defined by

$$K_0 = L, \quad K_{j+1} = \Omega(K_j), \quad j \geq 0$$

converges after a finite number of terms to $S = \sup\underline{C}(L)$. Furthermore, S is a regular language, with

$$\|S\| \leq \|L\| \cdot \|M\| + 1.$$

The following sections make use of some results and notation for which the reader may refer to [2] for the detailed explanation.

2. Procedure to Construct $C_0 = Tr(A \times B)$

In the applications so far considered, the language M is closed (i.e. $M = \bar{M}$) and represents the uncontrolled "physically possible" behavior of our discrete event process; while $L \subset M$ is the "legal" language describing behavior that is both possible and acceptable. For this special case we can compute $S = \sup\underline{C}(L)$ as follows (cf. [2], [3]).

Recall that a language $K \subset \Sigma^*$ is regular iff there exists a finite automaton

$$A = (Q, \Sigma, \delta, q_0, Q_m)$$

with

$$K = \{s : s \in \Sigma^* \ \& \ \delta(s, q_0) \in Q_m\}$$

The automaton A is often called a *recognizer* for K; we write $|A| = K$. The transition function $\delta : \Sigma \times Q \to Q$ is in general only a partial function, meaning that, for each $q \in Q$, $\delta(\sigma, q)$ is only defined for a subset $\Sigma(q) \subset \Sigma$ that depends on q. We call $\Sigma(q)$ the *active set* at the state q.

Now let $A = (Q, \Sigma, \delta, q_0, Q_m)$ and $C = (X, \Sigma, \xi, x_0, X_m)$ be trim (i.e. accessible and co-accessible, cf. [2]) finite automata with $|C| \subset |A|$. We say that C *refines* A if

$$\text{For all } s, t \in \overline{|C|}, \xi(s, x_0) = \xi(t, x_0) \text{ implies } \delta(s, q_0) = \delta(t, q_0)$$

If C refines A then there exists a unique function $h : X \to Q$ satisfying

$$h \circ \xi(s, x_0) = \delta(s, q_0), \quad s \in \overline{|C|}$$

Let

$$A = (Q, \Sigma, \delta, q_0, Q)$$

be a trim recognizer for M, and

$$B = (Z, \Sigma, \alpha, z_0, Z_m)$$

be a trim recognizer for L. As a preliminary to computing $\sup \underline{C}(L)$ we construct an automaton C_0 that is a recognizer for L and refines A. We have the following procedure.

```
procedure1;
  (* denote $Q = \{q1,...,qs\}$; $q0 = q1$; $Z = \{z1,...,zt\}$; $z0 = z1$ *)
   begin
      $x1 := (q1, z1)$; $X := \{x1\}$; $p := 1$; $n := 2$;
      (* find state set $X$ *)
      repeat
         (* calculate transition function $\xi$ *)
         for all $\sigma \in \Sigma$ do
            (* denote $xp = (qi, zj)$ *)
            if $\delta(\sigma, qi)$ and $\alpha(\sigma, zj)$ are defined then
               begin
                  $qi' := \delta(\sigma, qi)$; $zj' := \alpha(\sigma, zj)$;
                  if $(qi', zj') \in X$ then
                     (* denote $(qi', zj') = xk$ *)
                     $\xi(\sigma, xp) := xk$
                  else
                     begin
                        $xn := (qi', zj')$; $\xi(\sigma, xp) := xn$;
                        $X := X \cup \{xn\}$; $n := n + 1$
                     end
               end
            else $\xi(\sigma, xp)$ is undefined;
         $p := p + 1$
```

```
        until $n = p$;
        $x0 := x1$; $Xm := \{x = (q,z): x \in X, z \in Zm\}$
    end.
```

$C_0 = (X, \Sigma, \xi, x_0, X_m)$ is the recognizer for L that refines A.

The verification of procedure1 can be found in [6], where the complexity of procedure1 is discussed as well. It is found that the time required to perform procedure1 is $0(|Q|^2 \cdot |Z|^2 \cdot |\Sigma|)$ and the storage required is $0(|Q| \cdot |Z| \cdot |\Sigma|)$.

3. Procedure to Construct a Recognizer for $S = \sup \underline{C}(L)$

Let

$$A = (Q, \Sigma, \delta, q_0, Q)$$

be a trim recognizer for M and

$$C_0 = (X, \Sigma, \xi, x_0, X_m)$$

be the trim recognizer for L constructed by procedure1. We shall need the map $h: X \to Q$ such that

$$h \circ \xi(s, x_0) = \delta(s, q_0) \qquad s \in \bar{L}$$

A recognizer for $S = \sup \underline{C}(L)$ is constructed by the following procedure, which implements the iterative scheme described in Section 1.

```
procedure 2;
  begin
    $(X', \Sigma, \xi', x0, Xm') := (X, \Sigma, \xi, x0, Xm)$;
    (* compute $K(j+1) = \Omega(K(j))$ *)
    repeat
      $(X, \Sigma, \xi, x0, Xm) := (X', \Sigma, \xi', x0, Xm')$;
      $X' := \emptyset$;
      (* find state set $X'$ *)
      for all $x \in X$ do
        begin
          check:= 1
          (* check if $x \in X'$ *)
          for all $\sigma \in \Sigma u$ do
            if $\delta(\sigma, h(x))$ is defined then
              if $\xi(\sigma, x)$ is undefined then check:= 0;
          if check = 1 then $X' := X' \cup \{x\}$
        end
      (* calculate transition function $\xi'$ *)
      for all $x \in X'$, $\sigma \in \Sigma$ do
        begin
          $y := \xi(\sigma, x)$;
          if $y \in X'$ then $\xi'(\sigma, x) := y$
          else $\xi'(\sigma, x)$ is undefined
        end;
    until $X = X'$;
```

```
        $Xm' := X' \cap Xm$;
        $C := (X', \Sigma, \xi', x0, Xm')$
    end.
```

$C = (X', \Sigma, \xi', x_0, X_{m'})$ is the recognizer for sup $\underline{C}(L)$.

The verification of procedure2 is given in [6]. It is found that the time required to perform procedure2 is $0(|X|^3 \cdot |\Sigma|)$ and the storage required is $0(|X| \cdot |\Sigma| + |Q| \cdot |\Sigma|)$.

4. Program SUPCON

The foregoing procedures have been implemented in Pascal, and linked in the program SUPCON. The program listing can be found in [6].

5. Examples

In this section we illustrate the use of SUPCON with two examples. The first example is adapted from [2].

Example 1

A cat and a mouse are placed in the maze shown in Figure 1 with the cat initially in room 2 and the mouse initially in room 4. Each doorway in the maze must be traversed in the direction indicated and is either for the exclusive use of the cat (displayed as —| ↑ |—) or for the exclusive use of the mouse (displayed as —) ↑ (—). In addition each door, with the exception of c_7, can be opened or closed as required in order to control the movement of the cat and the mouse.

Our objective is to find the control scheme which permits the cat and the mouse the greatest possible freedom of movement but which also guarantees that

a) the cat and the mouse never occupy the same room simultaneously, and that

b) it is always possible for the cat and the mouse to return to the initial state, i.e. the state in which the cat is in room 2 and the mouse is in room 4.

Let $\Sigma = \{c_i, m_j : 1 \le i \le 7, 1 \le j \le 6\}$. We model the movement of the cat in the maze by the generator G_c over Σ shown in Figure 2, where state i corresponds to room i and a transition $i \xrightarrow{c_k} j$ corresponds to traversing the door c_k between rooms i and j.

Similarly we model the movement of the mouse in the maze by the grammar G_m shown in Figure 3.

Now for our joint model of the cat and the mouse we adopt the generator G constructed by shuffling the generators G_c and G_m. Denote $M = |G|$.

To construct a recognizer for the legal language L we remove those states (i, j) of G for which $i = j$, and take the trim component of the resultant automaton. Clearly this recognizer satisfies constraint (a). To satisfy constraint (b) we must ensure that every state is coaccessible to the state 24. Hence we designate 24 as the only marker state and again take the trim component of the resultant automaton. These computations yield a recognizer for L with 18 states as shown in

Figure 4.

Having specified M, L and Σ_0 we use the program SUPCON to compute the recognizer for $S = \sup \underline{C}(L)$, as displayed in Figure 5.

Example 2

This example illustrates a control problem for a computer communication network; for background see for instance [4], [5].

The network consists of two stations. Each station has a transmitter T and a receiver R. The two stations exchange messages through media M_1 and M_2. The messages are sent in "stop-and-wait" fashion (cf. [4]), i.e., T_1 sends a message and waits for acknowledgment by R_2. Only after such acknowledgment is received can T_1 send a new message. If a message arrives at R_2 while R_2 is busy, then the message will be stored in a buffer until R_2 is free and fetches it from the buffer. T_2 and R_1 work in the same way.

The state diagrams of T_i and $R_i (i=1,2)$, and the system signal flow are shown in Figure 6, where M_1 and M_2 are communication media.

Here $\alpha_i, \beta_i, \gamma_i, \sigma_i, \lambda_i, \mu_i$ stand for the following events.

α_i : transmitter starts sending a message

β_i : transmitter completes sending a message

γ_i : transmitter receives an acknowledgment

σ_i : receiver receives a message

λ_i : receiver starts sending an acknowledgment

μ_i : receiver completes sending an acknowledgment

The subset of controllable events is $\{\alpha_1, \alpha_2, \sigma_1, \sigma_2\}$.

The generator which describes the free (uncontrollable) system behavior is taken to be the shuffle grammar G of T_1, T_2, R_1, R_2; the language generated by G is denoted by M.

Because T_1 and R_1 (T_2 and R_2) cannot send a message or an acknowledgment through M_1 (M_2) at the same time, the recognizer for the legal language is obtained by removing the states in G in which both T_1 and R_2 (T_2 and R_2) are sending a message or acknowledgment.

By inspection, the legal language L is not controllable. We wish to find the supremal controllable sublanguage $S = \sup \underline{C}(L)$.

By use of program SUPCON, a recognizer for $S = \sup \underline{C}(L)$ was computed to have 49 states, while the recognizers for M and L have 81 and 64 states respectively. The detailed results can be found in [6]. The example can readily be extended to cases where the network has more than 2 stations.

6. Conclusions

In this report an algorithm has been developed to compute the supremal controllable sublanguage of a given language in the case where the languages involved are regular. An implementation is provided in Pascal, and two examples are presented for illustration.

7. References

[1] P.J. Ramadge and W.M. Wonham, Supervisory control of a class of discrete-event processes. In A. Bensoussan, J.L. Lions (eds.), *Analysis and Optimization of Systems*, LNCIS vol. 63, Springer-Verlag, 1984; Part 2, pp. 477-498.

[2] W.M. Wonham and P.J. Ramadge, On the supremal controllable and sub-language of a given language. *Proc. 23rd IEEE Conf. on Decision and Control*, IEEE Control Systems Society, December 1984, pp. 1073-1080.

[3] F. Lin, *Supervisor Synthesis for Discrete Event Processes*. M.A.Sc. Thesis, Dept. of Electrical Engineering, University of Toronto, 1984.

[4] A. Tannenbaum, *Computer Networks*. Prentice Hall, 1981.

[5] B.T. Hailpern and S.S. Owicki, Modular verification of computer communication protocols. *IEEE Trans. on Communication COM-31*, January 1983, pp. 56-68.

[6] F. Lin and W.M. Wonham, *On the computation of supremal controllable sublanguages*. Systems Control Group Rpt. #8506, Dept. of Electrical Engineering, University of Toronto, April 1985.

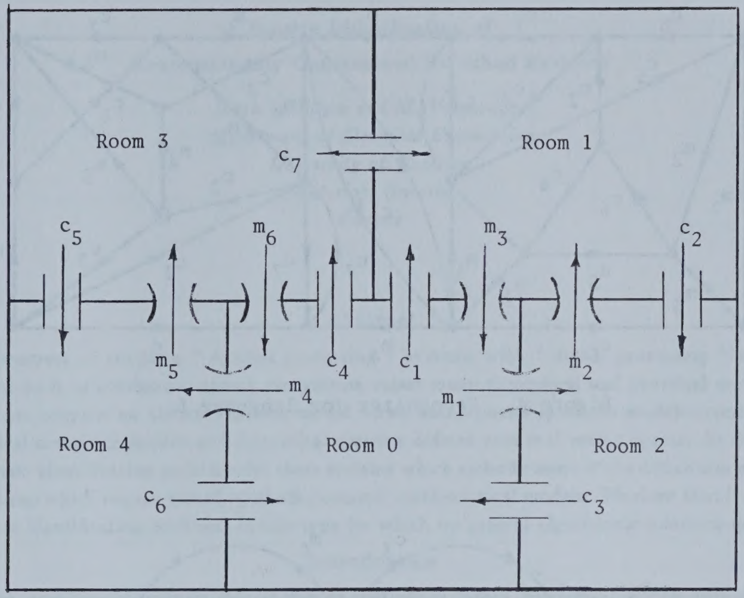

Room 3
c7
Room 1
c5
m6
m3
c2
m5
c4
c1
m2
m4
m1
Room 4
Room 0
Room 2
c6
c3

Figure 1 Maze for cat and mouse

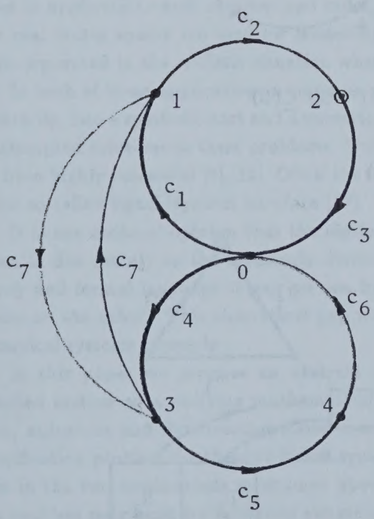

c2
1
2
c1
c3
c7
c7
0
c4
c6
3
4
c5

Figure 2 Movement of cat

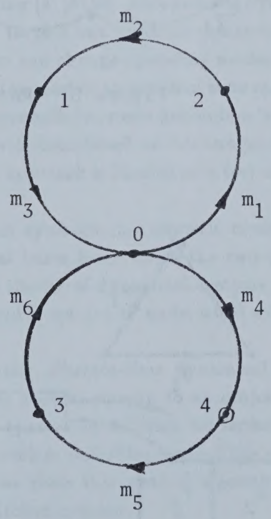

m2
1
2
m3
m1
0
m6
m4
3
4
m5

Figure 3 Movement of mouse

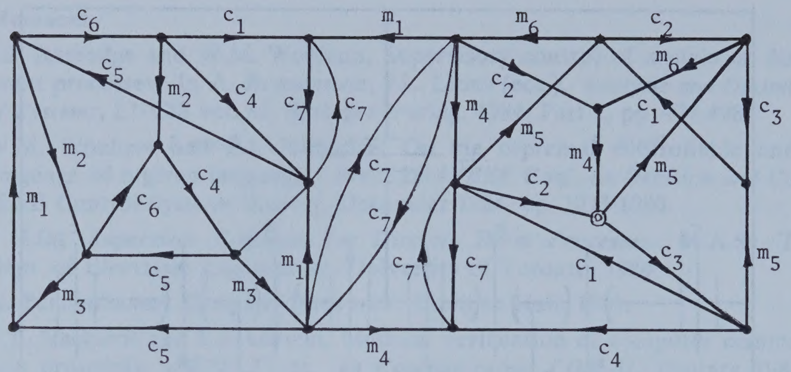

c_6
c_1
m_1
m_6
c_2
c_5
m_2
c_4
c_7
c_7
m_4
c_2
m_6
c_1
c_3
m_2
m_5
m_4
m_5
m_1
c_6
c_4
c_7
c_7
c_2
m_1
c_7
c_7
c_1
c_3
m_5
c_5
m_3
m_3
c_5
m_4
c_4

Figure 4 Recognizer for language L

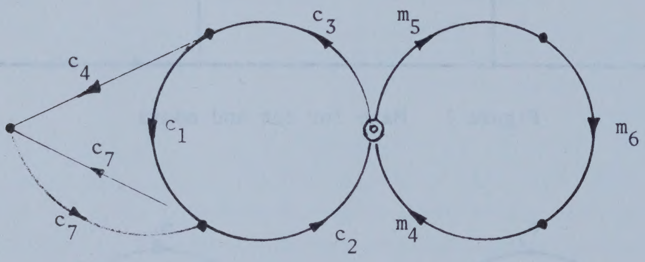

c_4
c_3
m_5
c_1
m_6
c_7
c_7
c_2
m_4

Figure 5 Recognizer of sup $\underline{C}(L)$

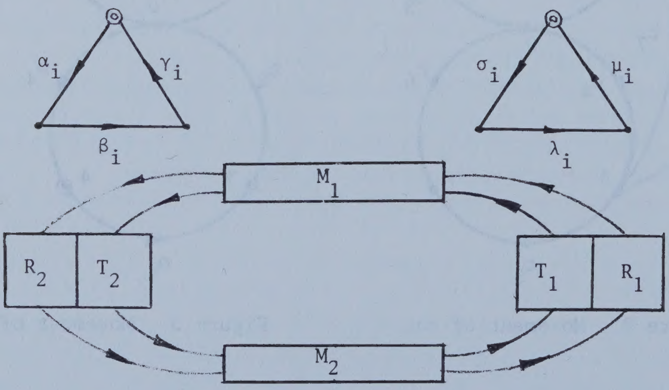

α_i
γ_i
β_i
σ_i
μ_i
λ_i
M_1
R_2
T_2
T_1
R_1
M_2

Figure 6 Computer communication network

Passive Identification of Grammatically Constrained Switched Systems

Mark McIntyre and M. Vidyasagar
Department of Electrical Engineering
University of Waterloo
Waterloo, Ontario
Canada

Abstract

The prospect of coupling " symbol processing " systems with " signal processing " systems in applications such as continuous speech recognition raises many theoretical and practical questions. In this paper we propose an abstract system model which encompasses symbolic models commonly used in theoretical computer science and dynamical systems defined over real vector spaces. In addition we formulate two identification problems for these systems which embody some of the difficulties which arise in applications which require mixed symbolic/numeric mathematical models. We show that it is possible to formulate identification problems of this type for which no general algorithmic solutions exist.

Introduction

For the modelling of complex signals such as continuous speech, dynamical models defined over real vector spaces are not adequate. While simple linear dynamical models are appropriate for primitive speech sounds [19], the grammatical constraints of a language require symbolic dynamical models which have been developed in automata theory and formal language theory [3],[16].

A requirement for dynamical models which incorporate both numeric and symbolic information also arises in applications such as sonar and radar target tracking [4],[5],[6]. Autonomous dynamical systems over real vector spaces are used for modelling stationary targets but symbolic dynamical models must be incorporated in the realistic situation where the targets can change operating modes abruptly.

In both of these applications a common approach to the modelling problem is to split it, somewhat arbitrarily, into a symbolic part and a numeric part. This symbolic/numeric distinction is carried through in attempted solutions to these problems. Unfortunately, solutions based on this two part approach have not been highly successful [6],[12]. Often the failure of this approach is blamed on a lack of understanding of the so-called signal/symbol interface [17].

It is the authors' opinion that the distinction between symbolic and numeric modelling which has arisen is due mainly to the seemingly distinct theoretical bases from which the two stem: automata theory and formal language theory on one hand and the theory of dynamical systems over real vector spaces on the other. This theoretical gap must be bridged if we are to understand symbolic/numeric dynamical systems properly.

In this paper we propose an abstract nonautonomous, discrete-time dynamical model called a switched system as a unifying mathematical model. It is general enough to encompass formal grammars, automata and dynamical systems over real vector spaces. In addition we formulate the passive identification problem for these switched systems. This problem embodies some of the difficulties which arise in the two applications mentioned above. Finally, we show that general algorithmic solutions to this problem only exist for relatively simple classes of switched systems.

The following notational conventions are followed in the paper: Capital letters such as X, U represent sets while lower case letters x, u represent elements of X, U respectively. Lower case bold letters $\mathbf{u}, \mathbf{t}$ represent elements of U^*, T^*, the free monoids over base alphabets U and T. Upper case bold letters $\mathbf{U}, \mathbf{T}$ are used to represent sets of strings or languages. Greek letters are reserved for denoting maps. Lower case Greek letters ψ denote single maps while upper case letters Ψ represent sets of maps. The notation X^U represents the set of all maps from the set U to the set X. Lastly, sequences in X^N are denoted by $\{\mathbf{x}\}$.

The Switched System Model

A straightforward way to describe a switched system is in terms of a nonautonomous difference equation.

Definition 1. *A (discrete-time) switched system is one which can be modelled by a nonautonomous difference equation*

$$x_{k+1} = \psi(x_k, u_k) \tag{1}$$

where $x_k \in X$, $u_k \in U$ for all $k \in \mathbb{N}$. The set X is an arbitrary state set, U is a finite input alphabet, k is a discrete-time index and $\psi(\cdot,\cdot) \; : \; X \times U \to X$.

Clearly, a switched system is distinguished from a general nonautonomous discrete-time system by the fact that the input alphabet is finite. Note that we do not distinguish between a switched system and a difference equation of the form (1) which models it.

A switched system can also be characterized by a triple $\langle X, U, \Psi \rangle$ where Ψ is a finite set of maps in X^X defined by

$$\Psi = \{\psi(\cdot, u) : \; u \in U\} \tag{2}$$

Note that X^X represents the set of mappings from X to itself. Implicit in this characterization is the function $\gamma(\cdot) : \; U \to X^X$ defined by

$$\gamma(u) = \psi(\cdot, u). \tag{3}$$

This map identifies symbols in the input alphabet U with maps in Ψ . Without loss of generality it can be assumed that $\gamma(\cdot)$ is one to one and U can be thought of as a set of distinct names for maps in Ψ.

A block diagram representation of a switched system based on this interpretation is shown below in Figure 1.

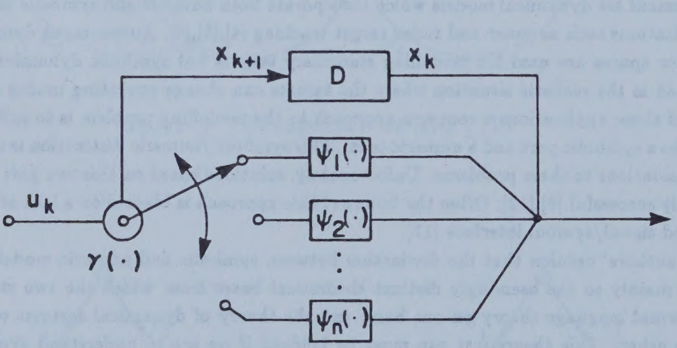


Figure 1 : A Switched System

In Figure 1, the block labelled $\mathcal{D}$ represents a unit time delay and $\gamma(\cdot)$ represents an input controlled switch which identifies symbols in U with functions in $\Psi = \{\psi_1(\cdot), \psi_2(\cdot), \ldots, \psi_n(\cdot)\}$. Note that $\mid U \mid = \mid \Psi \mid = n$ where $\mid \cdot \mid$ represents the cardinality of a finite set.

We define the *causal extension* of $\gamma(\cdot)$ to $\gamma^*(\cdot) : \; U^* \to \Psi^*$ by induction,

$$\gamma^*(\mathbf{e}) = I \tag{4}$$

$$\gamma^*(u \cdot \mathbf{u}) = \gamma(u) \circ \gamma^*(\mathbf{u}) \tag{5}$$

for all $u \in U$ and $\mathbf{u} \in U^*$. Here U^* denotes the free monoid with base alphabet U and identity $\mathbf{e}$ and ' $\cdot$ ' represents symbol concatenation. Also, Ψ^* denotes the monoid of maps generated by the maps in

Ψ under left composition ' $\circ$ '. The identity map in Ψ^* is denoted I. The function $\gamma^*(\cdot)$ takes strings of length n in U^* into maps in Ψ^* which in turn take states at some initial time k to states at time $k+n$. Note that $\gamma^*(\cdot)$ is an epimorphism from U^* to Ψ^*.

Based on $\gamma^*(\cdot)$ we define the map $\psi^*(\cdot,\cdot): X \times U^* \to X^+$ by

$$\psi^*(x, \mathbf{e}) = x \tag{6}$$

$$\psi^*(x, \mathbf{u}) = \prod_{j=0}^{n} (\gamma^*(T_0^j(\mathbf{u})) \mid_x) \tag{7}$$

for all $x \in X$, $\mathbf{u} \in U^*$. Here $\prod_0^n$ represents concatenation of n symbols and $T_l^m(\cdot): \mathbb{N} \times \mathbb{N} \times U^* \to U^*$ is defined by

$$T_l^0(\mathbf{u}) = \mathbf{e} \tag{8}$$

$$T_l^m(u_1 u_2 \dots u_n) = u_l u_{l+1} \dots u_{l+m-1} \tag{9}$$

for all $\mathbf{u} \in U^*$ such that $\mid \mathbf{u} \mid \geq l+m$, (Here $\mid \mathbf{u} \mid$ represents the length of a string). Clearly, $\psi^*(\cdot,\cdot)$ is an initial state dependent map which takes input strings to strings of states in X^+, the free semigroup generated by the symbols in X. This characterisation of the dynamic behaviour of a switched system is useful in formalizing the passive identification problem.

Relationship to Other Common Models

The switched system model is closely related to other mathematical models which appear in the literature on automata theory, formal language theory and discrete-time dynamical systems.

First, the finite automaton is a common model which is used in theoretical computer science. A closely related model which can be more easily related to a switched system is the semiautomaton [9].

Definition 2. *A (completely specified, deterministic, finite) semiautomaton is characterised by a triple $\langle Q, L, \Delta \rangle$ where Q, L are finite state and input alphabets respectively and $\Delta = \{\delta_l : l \in L\}$ is a set of maps in Q^Q.*

As is the case with switched systems, a semiautomaton undergoes a state transition in response to an input symbol $l \in L$. The transition is determined by the map δ_l. Clearly, the map $\delta(\cdot,\cdot): Q \times L \to Q$ defined by

$$\delta(q, l) = \delta_l \mid_q \tag{10}$$

defines a switched system. Moreover the switched system model can be thought of as a generalization of the semiautomaton model to the case where the state set is not necessarily finite.

A finite automaton can be defined in terms of a semiautomaton.

Definition 3. *A finite automaton is characterised by a quintuple $\langle Q, L, \Delta, q_0, F \rangle$ where $\langle Q, L, \Delta \rangle$ is a finite semiautomaton, $q_0 \in Q$ is a distinguished initial state and $F \subset Q$ is a distinguished subset called the final states.*

Usually finite automata are regarded as string recognition devices. A string $\mathbf{l} \in L^*$ is recognized or accepted by an automaton $\langle Q, L, \Delta, q_0, F \rangle$ if

$$\gamma^*(\mathbf{l}) \mid_{q_0} = q \in F \tag{11}$$

where $\gamma^*(\cdot)$ is defined naturally as in (4),(5). Regarded in this way, a finite automaton simply computes a function from L^* to $\{0, 1\}$. The dynamic evolution of the states is not of primary importance as it is in switched systems.

We now establish a connection between switched systems and formal grammars. Unlike automata which accept sets of strings, formal grammars are schemes for generating sets of strings over some alphabet. Unfortunately, it appears to be difficult to interpret these generation schemes as deterministic procedures. A closely related concept is that of a Markov algorithm [1]. These are formal symbol manipulation procedures which can be considered as deterministic versions of formal grammars [15]. Like formal grammars, Markov algorithms are defined in terms of productions.

Definition 4. *A production p over a base alphabet M is an ordered pair $(\mathbf{m}_1, \mathbf{m}_2) \in M^* \times M^*$. $\mathbf{m}_1$ is called the left hand side (LHS) of the production while $\mathbf{m}_2$ is called the right hand side (RHS).*

A production p is applied to a string $\mathbf{m} \in M^*$ by replacing one occurence of the LHS of p in $\mathbf{m}$ by the RHS of p.

A Markov algorithm is defined by a set of productions together with a convention for deciding where productions should be applied to strings.

Definition 5. *A Markov algorithm over an alphabet M is a finite set of productions $P \subset M^* \times M^*$ which are applied to strings according to the following rule. If the LHS of a production matches more than one substring of a string $\mathbf{m} \in M^*$ then the production is applied to the left-most occurence of the substring.*

By including the convention of choosing the left-most occurence of a substring match, application of a production $p \in P$ to a string $\mathbf{m} \in M^*$ yields a unique result. Such an application of a production to a string will be denoted $\mathbf{m} \xrightarrow{p} \mathbf{m}'$.

For a Markov algorithm with production set P over base alphabet M a corresponding switched system can be defined in a natural way. Consider the set $X = M^* \cup \#$ where $\#$ is a distinguished symbol not in M. Each $p \in P$ defines a function $\psi(\cdot, p) : X \to X$ according to the rules,

$$\psi(x,p) = \begin{cases} x' & \text{if } x \in M^* \text{ and } x \xrightarrow{p} x' \\ \# & \text{if } x \in M^* \text{ but } p \text{ cannot be applied to } \mathrm{x}. \end{cases}$$

$$\psi(\#, p) = \# \quad \text{for all} \quad p \in P$$

By considering each production in P a function $\psi(\cdot,\cdot) : X \times P \to X$ is defined. This map characterizes a switched system with state set $M^* \cup \#$ and input set P corresponding to the individual productions of the Markov algorithm.

As a final comparison the relationship between switched systems and nonautonomous, discrete-time dynamical systems is established. Such systems are sometimes called discrete processes [14].

Definition 6. *A map $\pi(\cdot,\cdot,\cdot) : \mathbb{N} \times \mathbb{N} \times X \to X$ is called a discrete process on X if,*
(i) $\pi(0, j, x) = x$ *for all* $j \in \mathbb{N}, x \in X$
(ii) $\pi(n, j+k, \pi(k, j, x)) = \pi(n+k, j, x)$ *for all* $j \in \mathbb{N}$ *for all* $n, k \in \mathbb{N}$ *and for all* $x \in X$
(iii) $\pi(\cdot,\cdot,\cdot)$ *is continuous.*

To each switched system there corresponds a naturally defined discrete process. For a switched system $\langle X, U, \Psi \rangle$, the map $\pi(\cdot,\cdot,\cdot) : \mathbb{N} \times \mathbb{N} \times X \to X$ defined by

$$\pi(n, j, x) = \gamma^*(T_j^k(\{\mathrm{u}\})) \mid_x \tag{11}$$

where $\{\mathrm{u}\} \in U^{\mathbb{N}}$ and where $T_j^n(\cdot) : \mathbb{N} \times \mathbb{N} \times U^{\mathbb{N}} \to U^*$ is defined as in (8),(9). It is straightforward to show that $\pi(\cdot,\cdot,\cdot)$ satisfies conditions (i) and (ii) in Definiton 6. Condition (iii) is also straightforward if we consider the discrete topology on X. Note that the extension of $\psi^*(\cdot,\cdot)$ defined in (6),(7) to a map $\psi^*(\cdot,\cdot) : X \times U^{\mathbb{N}} \to X^{\mathbb{N}}$ is an initial state dependent map which takes sequences of input symbols to sequences of states which are solutions to (1).

The Passive Identification Problem

The following general description of the dynamic system identification problem is due to Zadeh [21].

> " Identification is the determination on the basis of input and output, of a system within a specified class of systems to which a system under test is equivalent."

This description is taken as basic in a well known survey paper on system identification [2]. Implicit in the above statement of the passive identification problem is the assumption that inputs to some system under test are available for application of test signals. In applications such as continuous speech recognition and passive sonar and radar tracking this assumption is not valid. In the spirit of Zadeh's description we define the general passive identification problem.

Definition 7. *The passive identification problem is the determination on the basis of output alone, of a system within a specified class of systems to which a system under test is equivalent.*

From this definition it follows that two things must be specified in order to formalize this problem. First, the class of systems under test must be specified. In the passive identification case this specification must include assumptions regarding the types of allowable inputs. Second, some notion of equivalence or indistinguishability between systems must be specified.

There are two classes of systems in which we are interested.

Definition 8. *A grammatically constrained switched system is a switched system* $\langle X, U, \Psi \rangle$ *which can only be driven by input strings which belong to some proper sublanguage* $\mathbf{U} \subset U^*$.

Grammatically constrained switched systems will be denoted by triples $\langle X, \mathbf{U}, \Psi \rangle$ where $\mathbf{U}$ is the language to which input strings must belong. The second class of systems in which we are interested is the class consisting of grammatically unconstrained switched systems.

Definition 9. *A grammatically unconstrained switched system is a switched system which can be driven by any string* $\mathbf{u} \in U^*$.

Grammatically unconstrained switched systems will be denoted by pairs $\langle X, \Psi \rangle$. Note that since U and Ψ are in one to one correspondence, specifying both is redundant.

For the purposes of formulating the passive identification problem we shall assume that a system to be identified belongs to some class of switched systems denoted by $\mathcal{S}$. Individual systems within the class will be denoted by s and the system to be identified will be denoted by $\hat{s}$. If $\mathcal{S}$ is a class of grammatically constrained switched systems then each $s \in \mathcal{S}$ is characterized by a triple $\langle X_s, \mathbf{U}_s, \Psi_s \rangle$.

Definition 10. *Two switched systems* s_1, s_2 *in a class of grammatically constrained switched systems* $\mathcal{S}$ *are said to be equal if the triples which characterise them are equal.*

Equality of two grammatically unconstrained switched systems is defined in a similar way.

The passive identification problem can be roughly stated as follows: Based on a set of measurements of the state of a switched sytem $\hat{s}$ we would like to know which systems $s \in \mathcal{S}$ could have generated the set. If two systems $s_1, s_2 \in \mathcal{S}$ could have generated the set of measurements then the two systems are indistinguishable with respect to that set. Clearly, equality of two systems implies indistinguishability but the converse is not true.

In order to complete the formalisation of the passive identification problem the notion of system indistinguishability must be formalized. To do this assumptions must be made regarding the following issues:

(i) Availability of information regarding the class of systems $\mathcal{S}$ to which some system under test belongs.
(ii) Whether or not the full system state is available for observation.
(iii) Whether or not full information regarding the state evolution is available.
(iv) Whether the observations are made sequentially on a single copy of $\hat{s}$ or in parallel on multiple copies of $\hat{s}$.

Various combinations of these assumptions lead to formalizations of different passive identification prolems similar to the various automaton identification problems presented in [20]. In order to clarify these ideas we present an example of the passive identification problem taken from the formal language theory literature.

Example 1. *The Grammatical Inference Problem.*

The grammatical inference problem is usually stated with reference to phrase structure grammars.

Definition 11. *A phrase structure grammar g is characterised by a four-tuple $\langle T, N, P, n_0\rangle$, where T is an alphabet of terminal symbols, N is an alphabet of nonterminal symbols, n_0 is a distinguished nonterminal symbol called the start symbol and P is a set of productions on $N \cup T$.*

The language generated by g, denoted $\mathcal{L}(g) \subseteq T^*$ is defined by,

$$\mathcal{L}(g) = \{\mathbf{t} : n_0 \overset{\mathbf{p}}{\Rightarrow} \mathbf{t},\ \ \mathbf{p} \in P^*\}. \tag{12}$$

Here $n_0 \overset{\mathbf{p}}{\Rightarrow} \mathbf{t}$ denotes that n_0 derives $\mathbf{t}$ by application of a string of productions $\mathbf{p}$.

A straightforward formulation of the grammatical inference problem is presented in [7]. Given a class of grammars $\mathcal{G}$ with the same terminal alphabet T and a set of strings $\mathbf{T} \subset T^*$ we must determine which grammars in $\mathcal{G}$ can generate this set of strings. In other words, for which grammars $g \in \mathcal{G}$ is $\mathbf{T} \subseteq \mathcal{L}(g)$?

We would like to be able to formalize this problem as a passive identification problem for a class of switched systems but it is not possible to directly relate a class of switched systems to a class of phrase structure grammars. There are two main reasons behind this. First, in a phrase structure grammar there is a distinguished global initial state and a distinguished set of final states. Second, there is no convention for applying productions to strings in a deterministic way. The first distinction causes no real difficulty but some deterministic convention for production application must be included if we are to state the grammatical inference problem in the switched system framework. It is known [11] that including a convention of applying productions to strings in a leftmost fashion does not affect the language generation capabilities of context-free grammars.

With the above convention included it follows directly that we can construct a switched system representation of any phrase structure grammar. To each grammar $g = \langle T, N, P, n_0\rangle$ there corresponds a switched system $s = \langle X, P, \Psi\rangle$ where $X = (T \cup N)^* \cup \#$ and Ψ is a set of maps in X^X constructed from the individual productions in the grammar. Note that P can be thought of as a set of names for the productions in the grammar. Now we can define $\mathcal{L}(g)$ in terms of the corresponding switched system s,

$$\mathcal{L}(g) = \{\mathbf{t} :\ \gamma^*(\mathbf{p})\,|_{n_0} = \mathbf{t},\ \ \mathbf{p} \in P^*\ , \mathbf{t} \in T^*\}. \tag{13}$$

Here, $\gamma^*(\cdot)$ is as defined in (4),(5). The set of strings $\mathbf{P} \subseteq P^*$ which satisfies the constraints of (13) specifies a grammatically constrained switched system $\langle X, \mathbf{P}, \Psi\rangle$ which in turn specifies the language $\mathcal{L}(g)$.

The grammatical inference problem can now be formalized as a passive identification problem.

Definition 12. *Consider a class of grammatically constrained switched systems S corresponding to a class of grammars $\mathcal{G}$ with terminal alphabet T and start symbol n_0. Two systems $s_1, s_2 \in S$ are said to be indistinguishable with respect to a set of strings $\mathbf{T} \subset T^*$ if for each $\mathbf{t} \in \mathbf{T}$ there exists $\mathbf{p}_1 \in \mathbf{P}_1, \mathbf{p}_2 \in \mathbf{P}_2$ such that*

$$\gamma_1^*(\mathbf{p}_1)\,|_{n_0} = \mathbf{t} = \gamma_2^*(\mathbf{p}_2)\,|_{n_0} \tag{14}$$

The problem of determining whether or not two systems s_1, s_2 are indistinguishable with respect to a set of strings $\mathbf{T}$ is called the indistinguishability problem and is denoted $I(s_1, s_2, \mathbf{T})$.

Based on Definition 12 a version of the passive identification problem which corresponds to the grammatical inference problem can be formalised.

Problem 1. *Suppose we have full knowledge regarding the class of systems S, (with global initial state n_0), to which some system under test $\hat{s}$ belongs. Also suppose we have as measurements a set of strings $\mathbf{T} \subset T^*$ generated by parallel copies of $\hat{s}$ according to,*

$$\gamma_{\hat{s}}^*(\mathbf{p})\,|_{n_0} = \mathbf{t} \tag{15}$$

for some $\mathbf{p} \in \hat{\mathbf{P}}$. *The passive identificaton problem is one of determining which system(s)* $s \in \mathcal{S}$ *are indistinguishable from* $\hat{s}$ *with respect to the set of measurements* $\mathbf{T}$.

In this problem statement a number of assumptions of the type mentioned earlier have been made. First we have assumed that the system under test belongs to some class which has been specified a priori. This excludes the possibility that the set of measurements could have been generated by a system which is not in $\mathcal{S}$. Second, it is clear that no information regarding the state evolution is available and only terminal states are observable. Lastly it is assumed that the observable strings are generated independently by parallel copies of $\hat{s}$ all started in state n_0. These assumptions should be contrasted with those made in the next example.

Example 2. *Realisation of Discrete-Time Linear Systems from Their Pulse Response.*

The problem of computing a state space realisation of a linear, time-invariant, discrete-time dynamical system from measurements of the pulse response of the system is a well known and well studied problem in the System Theory literature [10],[18]. A simplified version of this problem can be easily stated in the switched system framework.

Consider an autonomous linear system,

$$x_{k+1} = \psi(x_k) \tag{16}$$

where $x_k \in \mathbb{R}^n$ for all $k \in \mathbb{N}$ and $\psi(\cdot)$ is a vector space homomorphism. The realization problem is one of finding a representation for the map $\psi(\cdot)$ in (16) based on observations of a solution sequence $\{\mathbf{x}\}$. Note that (16) is just a special case of the difference equation (1). In this case $X = \mathbb{R}^n$ and the input alphabet consists of a single letter. That is, an autonomous linear system can be represented by a switched system $\langle X, \psi \rangle$ where X is a vector space and ψ is a single vector space homomorphism.

If we consider the class of systems,

$$\mathcal{S} = \{\langle X, \psi_s \rangle : X = \mathbb{R}^n, \psi_s \quad \text{is a homomorphism}\}$$

the realization problem for systems described by (16) can be stated as a passive identification problem. Given a sequence of measurements $\{\hat{\mathbf{x}}\} \in \mathbb{R}^N$ generated by a system under test $\hat{s}$ according to,

$$\hat{x}_n = \hat{\psi}^n(\hat{x}_0) \tag{17}$$

the realization problem is one of determining which systems $s \in \mathcal{S}$ could have generated the observed sequence.

The assumptions regarding the nature of the available measurements are quite different from those made in the grammatical inference problem. The main distinction is that the complete system state is observable at each discrete time instant. Rather than observing terminal states only, we observe the complete state evolution of the system under test. We shall now state formally a sequential version of the passive identification problem.

Definition 13. *Consider a class of grammatically constrained switched systems* $\mathcal{S}$ *with state set* X *and a sequence* $\{\mathbf{x}\}$. *Two systems* $s_1, s_2 \in \mathcal{S}$ *are sequentially indistinguishable with respect to* $\{\mathbf{x}\}$ *if for all substrings* $\mathbf{x}$ *of* $\{\mathbf{x}\}$ *there exist strings* $\mathbf{u}_1 \in \mathbf{U}_1$ *and* $\mathbf{u}_2 \in \mathbf{U}_2$ *such that,*

$$\psi_1^*(x_0, \mathbf{u}_1) = \mathbf{x} = \psi_2^*(x_0, \mathbf{u}_2) \tag{18}$$

where $x_0 = T_0^1(\{\mathbf{x}\})$. *The problem of determining whether two systems* s_1, s_2 *are indistinguishable with respect to a sequence* $\{\mathbf{x}\}$ *is termed the sequential indistinguishability problem and is denoted* $SI(s_1, s_2, \{\mathbf{x}\})$.

Problem 2. *Suppose full knowledge is available regarding a class of grammatically constrained switched systems S to which some system under test $\hat{s}$ belongs. Also suppose a sequence of states $\{\hat{\mathbf{x}}\}$ generated by $\hat{s}$ is available for measurement. The sequential passive identification problem is one of determining which systems in S are sequentially indistinguishable from $\hat{s}$ with respect to $\{\hat{\mathbf{x}}\}$.*

The identification problem for grammatically unconstrained switched systems is defined in a similar way.

On the Existence of General Algorithmic Solutions

We would like to be able to pose solutions to the passive identification problem which can be implemented as algorithms. Unfortunately, if we make use of the formal notion of an algorithm which is commonly used in theoretical computer science, it is possible to formulate identification problems for which no general algorithmic solutions exist. Such problems are termed undecidable. In this section we discuss versions of the passive identification problem which can be easily shown to be undecidable. A nontrivial example of a passive identification problem which is known to be decidable is also presented.

First, consider the passive identification problem described in Problem 1. To simplify matters it is assumed that the class of systems S to which a system under test $\hat{s}$ belongs is a finite family of systems. With this simplification the task of finding algorithmic solutions to the identification problem reduces to the problem of finding algorithmic solutions to the indistinguishability problem $I(s_1, s_2, \mathbf{T})$ for individual systems s_1, s_2 and sets of measurements $\mathbf{T}$.

A straightforward approach to solving $I(s_1, s_2, \mathbf{T})$ is to first develop algorithms for determining whether or not a particular string $\mathbf{t} \in T^*$ could have been generated by a system $s_1 \in S$. If such algorithms can be found then a procedure for solving $I(s_1, s_2, \mathbf{T})$ can be developed simply by checking whether each system in S could have generated each string in $\mathbf{T}$.

It is well known that procedures exist for determining whether particular strings over some alphabet could have been generated by any phrase structure grammar with suitable terminal alphabet. But, for type-0 grammars, such procedures cannot necessarily be made into algorithms. That is, even the problem of determining whether $\mathbf{t} \in \mathcal{L}(g)$ is an undecidable problem if g is a type-0 grammar. From this result we can conclude immediately that attention must be restricted to switched systems which correspond to type-1,2,3 phrase structure grammars if we expect to find algorithmic solutions to $I(s_1, s_2, \mathbf{T})$.

Further complications arise when we try to make the procedure proposed earlier into an algorithm. An important question which arises is: For what families of systems S and sets of measurements $\mathbf{T}$ can we expect to find general halting conditions for the procedure? In other words, for what families of systems and sets of measurements can we expect to find general algorithmic solutions to $I(s_1, s_2, \mathbf{T})$? Unfortunately, such general algorithmic procedures do not exist even for families of systems corresponding to context-free grammars. Consider the following result.

Lemma 1. *[8] The problem of determining whether $\mathcal{L}(g_1) \subseteq \mathcal{L}(g_2)$ is an undecidable problem for arbitrary context-free grammars g_1, g_2.*

From this lemma it follows directly that there are classes of systems for which $I(s_1, s_2, \mathbf{T})$ is an undecidable problem.

Lemma 2. *For arbitrary switched systems s_1, s_2 corresponding to context-free grammars and for arbitrary sets of strings $\mathbf{T}$ the indistinguishability problem $I(s_1, s_2, \mathbf{T})$ is undecidable.*

Proof: Consider two context-free grammars g_1, g_2.Let s_1, s_2 be the switched systems which correspond to the grammars. Also, let $\mathbf{T} = \mathcal{L}(g_1)$. Now suppose a general algorithm for solving $I(s_1, s_2, \mathbf{T})$ exists. If the two systems are indistinguishable with respect to $\mathbf{T}$ then $\mathcal{L}(g_1) \subseteq \mathcal{L}(g_2)$ otherwise $\mathcal{L}(g_1) \not\subseteq \mathcal{L}(g_2)$. The existence of such a general algorithm contradicts Lemma 1.•

Note that Lemma 2 does not imply that algorithmic solutions cannot be found for specific families of context-free grammars.

An example of the sequential passive identification problem which is decidable is the so-called DOL sequence equivalence problem.

Problem 3. *Given a string* $\mathbf{m} \in M^*$ *and two homomorphisms* $\psi_1(\cdot), \psi_2(\cdot)$ *which take* M^* *to itself. Is it decidable whether* $\psi_1^n(\mathbf{m}) = \psi_2^n(\mathbf{m})$ *for all* $n \in I\!N$*?*

It is interesting to note that this problem was open for quite some time before it was shown to be decidable in [13].

Now consider two switched systems $s_1 = \langle X, \psi_1 \rangle, s_2 = \langle X, \psi_2 \rangle$ where $X = M^*$ and ψ_1, ψ_2 are single homomorphisms. It follows directly from the above result that $SI(s_1, s_2, \{\mathbf{x}\})$ is a decidable problem for any sequence $\{\mathbf{x}\}$.

Conclusions

The abstract switched system model appears to be a useful tool for unifying ideas in theoretical computer science and discrete-time dynamical systems theory. Moreover, the passive identification problem for this class of systems is an interesting theoretical problem which addresses some of the issues which arise in applications such as continuous speech recognition and passive sonar target tracking.

We have formulated two versions of the passive identification problem for grammatically constrained switched systems. The first problem is closely related to the grammatical inference problem from formal language theory. The second problem is related to the realization problem for discrete-time, linear dynamical systems. It is interesting to note that both of these problems can be viewed as special cases of the general passive identification problem for switched systems.

Clearly, the difficulty involved in finding algorithmic solutions to these problems depends on the class of systems under consideration S. We have shown that if each $s \in S$ is defined over the same countable state set, then we cannot expect to be able to find general algorithmic solutions to the problems. This result raises an issue which is not usually discussed in the dynamical system identification literature. Namely, it is posible to formulate identification problems for which no algorithmic solutions exist. In the literature much attention is focussed on proving convergence of various system identification procedures. At least for the cases discussed in this paper, it is possible to formulate identification problems for which no such convergence proof can be found.

References

[1] Aiserman, M.A., L.A. Gusev, L.I. Rozoner, I.M. Smirnova and A.A. Tal, *Logic, Automata and Algorithms* , Academic Press, New York, 1971.

[2] Aström, K.J. and P. Eykhoff , *System Identification-A Survey* , Automatica, Vol. 7, 1971, pp 123-162.

[3] Bahl, L.R., F. Jelinek and R.L. Mercer , *A Maximum Liklihood Approach to Continuous Speech Recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 2, Mar. 1983, pp 179-190.

[4] Bruneau, F. and R.R. Tenney, *Optimal Smoothing and Estimation for Hybrid State Processes* , Report no. LIDS-P-1269, Dept. of Elect. Eng., M.I.T., Cambridge, Mass., Jan 1983.

[5] Chang, C.B. and M. Athans, *State Estimation for Discrete Systems with Switching Parameters* , IEEE Trans. on Aerospace and Electronic Systems, Vol AES-14, No. 3, May 1978, pp 418-424.

[6] Chang, C.B. and J.A. Tabaczynski, *Application of State Estimation to Target Tracking* , IEEE Trans. on Automatic Control, Vol. AC-29, No.2, Feb. 1984, pp 98-110.

[7] Fu, K.S. and T.L. Booth, *Grammatical Inference: Introduction and Survey-Part I* , IEEE Trans. Systems Man and Cybernetics, Vol. SMC-5. No. 1, Jan 1975, pp. 95-111.

[8] Ginsburg, S., *The Mathematical Theory of Context-Free Languages* , McGraw-Hill, New York, 1966.

[9] Ginzburg, A., *Algebraic Theory of Automata* , Academic Press, New York, 1968.

[10] Ho B.L. and R.E. Kalman, *Effective Construction of Linear State-Variable Models From Input-Output Functions*, Proc. Third Allerton Conference, Urbana, Illinois, 1966.

[11] Hopcroft,J.E. and J.D. Ullman, ***Introduction to Automata Theory, Languages and Computation***, Addison-Wesley, Reading, Mass., 1979.

[12] Kopec, G.E., A.V. Oppenhiem and R. Davis, ***Knowledge-Based Signal Processing*** , Trends and Perspectives in Signal Processing, Vol. 2, No. 3, July 1982, pp 1-6.

[13] Kulik, K. and I. Fris, ***The Decidability of the Equivalence Problem for DOL Systems***, Information and Control, No. 35, 1977, pp. 20-39.

[14] LaSalle, J.P., ***The Stability of Dynamical Systems***, SIAM Philadelphia, 1976.

[15] Machtey, M. and P. Young, ***An Introduction to the General Theory of Algorithms*** , North Holland, New York, 1978.

[16] Myers, C.S. and S.E. Levinson, ***Speaker Independent Connected Word Recognition Using a Syntax-Directed Dynamic Programming Procedure*** , IEEE Trans. on Acoustics Speech and Signal Processing, Vol. ASSP-30, No. 4, Aug. 1982, pp 561-565.

[17] Nii, P. , E.A. Figenbaum, J.J. Anton and A.J. Rockmore, ***Signal-to-Symbol Transformation: HASP / SIAP Case Study*** , The AI Magazine, Spring 1982, pp 23-35.

[18] Rissanen, J., ***Recursive Identification of Linear Systems*** , SIAM J. Control, Vol. 9, No. 3, Aug 1971, pp. 420-430.

[19] Schroeder, M.R., ***Linear Prediction, Entropy and Signal Analysis*** , IEEE ASSP Magazine, July 1984, pp 3-11.

[20] Trakhtenbrot,B.A. and Y.M. Barzdin, ***Finite Automata, Behaviour and Synthesis***, North-Holland, New York, 1973.

[21] Zadeh, L.A., ***From Circuit Theory to Systems Theory*** , Proceedings IRE, Vol. 50, 1962, pp 865-865.

ON THE USE OF TEMPORAL LOGIC IN CONTROL THEORY

J.G. Thistle and W.M. Wonham
Systems Control Group
Dept. of Electrical Engineering
University of Toronto
Toronto, Ontario
CANADA M5S 1A4

ABSTRACT

Linear time temporal logic is applied to the verification of a control system. A formal language and a proof system, both based on those used in the verification of concurrent programs by Manna and Pnueli [1], are briefly described. A property of a simple control system is then formally verified by deducing a temporal logic expression for the property from a description of the system dynamics. The usefulness of temporal logic in control theory is discussed and topics for research are suggested.

1. Introduction

Temporal logic has recently been proposed as a means of verifying concurrent programs, operating systems and communications protocols [1-3], because it allows easy description of the way program variables change during execution. It has also been applied to the verification of digital hardware [4] and the synthesis of concurrent programs [5,6]. The special features of temporal logic and its suitability for the above problems suggest that it will be useful in control theory -- indeed, some work has already been done on control problems within this framework [7]. The present paper describes an application of temporal logic to the verification of a simple control system.

In section 2, we give a brief description of the formal language and proof system used. In section 3, we describe an example, due to Dijkstra, and its formal verification. Finally, in section 4 we make an appraisal of the usefulness of temporal logic for control theory, and give some suggestions for future research.

2. Introduction to the Formalism

In [8], we described a version of linear-time temporal logic and a proof system for that language. Both of these are adapted from those of Manna and Pnueli's system, designed for program verification [1].

The Language

Linear time temporal logic is similar to the predicate calculus, but its individual variable symbols -- called "local variable symbols" -- are interpreted as quantities that vary over an infinite sequence of discrete points in time. Formally, the formulas of the language are evaluated with respect to "structures", each consisting of an "interpretation" and a "sequence" of "states". The interpretation assigns a value to every symbol other than local variable symbols and logical symbols. Each state in the sequence assigns a value to every local variable symbol. Hence if the sequence is thought of as a sequence of instants in time, then the local variable symbols have values that vary with time, while the other symbols do not. Formulas containing these local variable symbols thus have truth values that can be thought of as varying in time. Special "temporal operators" describe the time-dependence of the formulas they act on, and this gives the language its power to describe time-varying quantities. These are some of the temporal operators and their intuitive interpretations:

$\bigcirc w$ - "the formula w will be true in the next instant";
$\Box w$ - "the formula w will be true henceforth";
$\Diamond w$ - "the formula w will be true eventually";
$v \,\mathbf{U} w$ - "the formula w will be true at some future instant, and v will be true until then".

Of course, formulas containing more than one temporal operator tend to have more complex meanings:

$\Diamond \Box w$ - "eventually, w will become permanently true" [1];
$\Box \Diamond w$ - "there will always be a future time at which w will be true", or w will be true infinitely often" [1].

The next operator, " $\bigcirc$ ", may also be applied to terms of the language: for any term t, the term ($\bigcirc$t) always has the value that t will have in the next instant.

The Proof System

We can think of structures as representing state trajectories of a dynamic system. Temporal logic formulas can then be used to give precise descriptions of these structures, and hence of the system itself. In the next section we will give such a temporal logic description of a system and then formally verify a property of its behavior.

The problem of verifying a discrete system is the following: given a description of the (one-step) dynamics of the system -- the way its state changes from one instant to the next -- one must show that the system satisfies a given property, which in general depends not only on the way its state changes from one instant to the next but on entire state trajectories.

Thus, in a formal verification we give temporal logic descriptions of the one-step dynamics and of the desired property of the system's behavior. The verification consists in showing that any structure satisfying the description of the system dynamics, and hence representing a physically possible trajectory of the system, also satisfies the formal statement of the desired property. For this purpose, we present in [8] a proof system for our language.

This proof system consists of a set of valid formulas that we take as axioms, and the inference rule "modus ponens". In [8], we list several derived rules -- schematic descriptions of inferences that can be made within the proof system. In the present paper we list some that will be of use in the example that follows. The first is called "Derived Computational Induction" (DCI):

$$\frac{\Box\,(w \Rightarrow \bigcirc w)}{\Box\,(w \Rightarrow \Box w)} \qquad \text{(DCI)}$$

The notation is standard: the connective " $\Rightarrow$ " represents conditional implication (we will use "$\wedge$" and "$\vee$" to represent conjunction and disjunction, respectively); the rule means that the formula appearing below the line can be deduced from that (those) appearing above it.

The premise simply means that whenever w is true, it will be true at the next instant. According to the rule, we can infer from this that if w ever becomes true, it will remain true from then on (an obviously valid inference).

DCI is essentially a simple corollary of a rule of Manna and Pnueli's of the same name. Our next rule is also adapted from one of theirs:

$$\begin{array}{l} \Box(w_1 \wedge \neg w_3 \Rightarrow \bigcirc w_1) \\ \Box(w_1 \wedge w_3 \Rightarrow \bigcirc w_2) \\ w_1 \Rightarrow \Diamond w_3 \\ \hline w_1 \Rightarrow (w_1 \mathbf{U} w_2) \end{array} \qquad \text{(RUI)}$$

This rule is called "Right 'Until' Introduction" because it describes the deduction of a formula in which the "until" operator appears on the right side of a conditional implication connective. The conclusion means that if w_1 is currently true, then w_2 will eventually become true, and w_1 will remain true until then. By the third premise, if w_1 is true, then w_3 will eventually become true, and the first premise implies that w_1 will remain true up to, and including, that instant. By the second premise, w_2 will become true at the next instant, so the conclusion is justified.

In applying RUI, we will infer the weaker conclusion that has $\Diamond w_2$ as the consequent of the conditional implication.

"Chain" is another rule adapted from one of Manna and Pnueli's of the same name. It describes a series of arguments of the form of RUI.

$$\begin{array}{l} \left.\begin{array}{l} \Box(v_i \wedge \neg w_i \Rightarrow \bigcirc v_i), \\ \Box(v_i \wedge w_i \Rightarrow \bigcirc \bigvee_{j=0}^{i-1} v_j), \\ \Box(v_i \Rightarrow \Diamond w_i) \end{array}\right\} \text{ for all } i, 0<i\le n \\ \hline \Box(\bigvee_{i=0}^{n} v_i \Rightarrow (\bigvee_{i=0}^{n} v_i)\,\mathbf{U}\, v_0) \end{array}$$

Here, any formula of the form $\bigvee_{i=m}^{n} w_i$ represents a disjunction, $w_m \vee w_{m+1} \vee \cdots \vee w_n$. Similar abbreviations will be used for conjunctions.

As with RUI, when invoking Chain we will infer the weaker conclusion that has $\Diamond v_0$ as the consequent of the conditional implication.

The above derived rules capture most of the temporal reasoning used in the example of the next section. However, as we will be using integers to describe the state of our control system, we will also need to do some mathematical reasoning. Manna and Pnueli include in their proof system a "domain part" which allows them to do such reasoning formally. For simplicity, we make the corresponding part of our system a degenerate one: for any formula w, containing no temporal operators, that is true under the obviously intended interpretation of its symbols, we simply define $\Box w$ to be an axiom. Now we can make all of the valid mathematical inferences that we need. In proofs, such steps will be described as being justified "by mathematical reasoning".

This completes our description of the formalism. We now illustrate its use in the verification of control systems.

3. Dijkstra's "Self-Stabilizing Distributed System" [9]

The System

Consider N+1 finite state machines, $m_0,\ldots,m_N$, arranged in a ring so that any machine m_n has as its neighbors $m_{(n-1) \bmod (N+1)}$ and $m_{(n+1) \bmod (N+1)}$. The state of any machine m_n is represented by an integer x_n, $0 \le x_n < K$, where K is some integer $\ge N$.

The possible state transitions, or "moves", and the conditions under which they are allowed to occur are the following:

(i) m_0 may increment the value of x_0 by 1 (mod K), as long as $x_0 = x_N$ initially;

(ii) any other machine m_i, $0 < i \le N$, can change x_i to equal x_{i-1}, if $x_i \ne x_{i-1}$ initially.

The machines make moves only when their enabling conditions are satisfied, and only one at a time.

Note that each machine communicates only with its neighbors: the outcome of a move made by any machine, and the enabling condition for that move, depend only on the initial state of that machine and those of its neighbors. In this sense the system is a distributed one.

The problem is to show that, no matter what its initial state, the system will eventually enter a set of "legitimate states" defined by a condition on the *global* system state -- this in spite of the fact that individual machines use only *local* information. Specifically, the legitimate states are those for which there exists an n, $0 \le n \le N$, such that

$$x_i = x_0, \quad \forall i, \quad 0 \le i \le n,$$

and

$$x_j = (x_0 - 1) \bmod \mathrm{K}, \quad \forall j, \quad n < j \le N.$$

In order to understand the dynamics of the system on the set of legitimate states, consider the case where $x_0,\ldots,x_N$ are all the same. Only m_0 is enabled, so in the next system state, x_0 will have been incremented by 1 (mod K) and $x_1,\ldots,x_N$ will be unchanged. Then only m_1 will be enabled, so x_1 will be incremented by 1 (mod K). Then m_2 will make the same move, and so on. Eventually, all of the machines will have made the same move and $x_0,\ldots,x_N$ will once again all be equal. The set of states visited during this cyclic operation of the system are exactly those we have defined to be legitimate states.

Dijkstra's legitimate set includes other states as well, but Dijkstra himself requires that the legitimate states be so defined that any one of them can be reached from any other. Clearly, those that belong to his legitimate set but not to ours cannot be reached from our legitimate states, which form an invariant subset. Hence, Dijkstra's set does not satisfy the requirement. This illustrates the fact that informal reasoning about even a simple system is unreliable.

It is not obvious that the system will, whatever its initial state, eventually enter the legitimate set, and Dijkstra does not show that this is so. We now outline a temporal logic proof.

Formal Verification

Recall that in order to verify formally a property of a system, we must show that any structure satisfying a temporal logic description of the system dynamics also satisfies a temporal logic expression of the desired property, and that for this it suffices to show that the latter can be deduced from the former within our proof system. Thus we now write down formal descriptions of the dynamics of Dijkstra's system and its property of eventually entering its set of legitimate states, and outline a proof.

In describing dynamic systems, it is convenient to be able to refer to events. We will call some of our constant symbols "event symbols" and use these to represent state transitions that the system can undergo. The local variable symbol Δ will, at any instant, stand for the next event that is to occur. Hence the formula $\Delta = \alpha$ means that the event represented by the event symbol α is about to occur.

The moves of $m_0,\ldots,m_N$ will be represented by the event symbols $\alpha_0, \ldots, \alpha_N$, and their states by the local variable symbols $x_1,\ldots,x_N$.

The following formulas constitute our description of the system and its one-step dynamics:

$$\bigwedge_{i=0}^{N} \bigvee_{j=0}^{K} x_i = j \tag{SS1}$$

This means that the state of each machine is initially one of the integers 0 through K-1.

$$\Box \bigvee_{i=0}^{N} \Delta = \alpha_i \tag{SS2}$$

This formula says that the only events that occur are the moves that the various individual machines can make. Furthermore, it says that these events continue to occur: it is not difficult to see that it is impossible for all of the machines to be disabled simultaneously, so we make this useful fact part of our basic description of the system.

$$\Box[\Delta=\alpha_0 \Rightarrow x_0=x_N \wedge (\bigcirc x_0)=(s_0+1) \bmod \mathrm{K} \wedge \bigwedge_{j=1}^{N} ((\bigcirc x_j)=x_j)] \tag{SS3}$$

This formula describes the effects of state transitions made by m_0, and the conditions under which such moves can occur: m_0 can make a move only if $x_0 \neq x_N$, and the effect is to increment the value of x_0 by 1 (mod K), leaving $x_1,\ldots,x_N$ unchanged.

$$\Box \bigwedge_{i=1}^{N} [\Delta=\alpha_i \Rightarrow x_i \neq x_{i-1} \wedge (\bigcirc x_i)=x_{i-1} \wedge \bigwedge_{\substack{j=0 \\ j\neq i}}^{N} ((\bigcirc x_j)=x_j)] \tag{SS4}$$

This says that any machine m_i, $0 < i \leq N$, can make a move only if $x_i \neq x_{i-1}$, with the effect of making x_i equal to x_{i-1}, leaving the states of all other machines unchanged.

This completes our formal description of the system.

Let γ stand for the followng formula, which is simply a formal expression of the condition defining the legitimate states:

$$\bigvee_{n=0}^{N-1} [(\bigwedge_{i=1}^{n} x_i = x_0) \wedge (\bigwedge_{j=n+1}^{N} x_j = (x_0 - 1) \bmod \mathrm{K})] \vee (\bigwedge_{j=1}^{N} x_j = x_0)$$

The formula $\Diamond\gamma$ expresses the desired property: it says that the system will eventually be in the legitimate set. The formal verification therefore consists in proving the following

Theorem

The formula $\Diamond\gamma$ can be deduced from SS1-SS4.

In [8] we give a detailed proof of this theorem. In the present paper, we will state the main intermediate results, explain their significance, and outline their proofs with informal arguments.

Lemma 1

The formula

$$\Box \bigwedge_{i=0}^{N} \bigvee_{j=0}^{K-1} x_i = j$$

can be deduced from SS1-SS4.

This simply means that the state of any machine is always one of the integers 0 through K-1. According to SS1, this is true initially, and the other specifications imply that its truth is preserved. Thus the formula follows from the specifications by DCI.

The next result expresses the fact that no machine ever permanently stops making moves -- each makes a move "infinitely often".

Lemma 2

$$\bigwedge_{i=0}^{N} \Box \Diamond \Delta = \alpha_i$$

can be deduced from SS1-SS4.

Lemma 2 is a straightforward consequence of the following

Claim

For any i, $0 \le i \le N$, the formula

$$\Diamond \Box \Delta \neq \alpha_i \Rightarrow \Diamond \Box \Delta = \alpha_{(i+1) \, mod \, (N+1)}$$

can be deduced from SS1-SS4.

This result means that if any machine m_i ever permanently stops making moves, then $m_{(i+1) \, mod \, (N+1)}$ will eventually do the same. This follows easily from SS3 and SS4: these two formulas imply that whenever any machine $m_{(i+1) \, mod \, (N+1)}$ makes a move, its enabling condition is falsified, and that this condition cannot be satisfied again until m_i has made a move. Hence, if m_i permanently stops making moves, then $m_{(i+1) \, mod \, (N+1)}$ can make at most one more.

Lemma 2 is now easy to prove: by the claim, if any machine stops making moves, then all must eventually stop. But this contradicts SS2. Hence all machines move infinitely often.

Lemmas 1 and 2 are basic results to be used in later stages of the proof. We now use the rest of our main results to make up a "backwards" proof of the theorem.

In order to deduce $\Diamond\gamma$, and prove the theorem, it suffices to deduce the weaker formula

$$\Diamond(\bigwedge_{j=1}^{N} x_j = x_0),$$

which means that the system will eventually enter a legitimate state of the particular type in which all of $x_0,\ldots,x_N$ are equal. Our next result states that this will in fact occur, if at some point none of $x_1,\ldots,x_N$ equals x_0. For this we make the following

Definition

Let ζ_n stand for

$$\begin{array}{ll} \bigwedge_{i=0}^{n} x_i = x_0 \wedge \bigwedge_{j=n+1}^{N} x_j \neq x_0, & \text{if } 0 \leq n < N, \\ \bigwedge_{i=1}^{N} x_i = x_0, & \text{if } n = N. \end{array}$$

Thus ζ_n means that $x_0,\ldots,x_N$ all have the same value, and that value is different from all of those of $x_{n+1},\ldots,x_N$. Our goal is to deduce $\Diamond\zeta_N$, and we use the intermediate result

Lemma 3

The formula

$$\Box[\zeta_0 \Rightarrow \Diamond\zeta_N]$$

can be deduced from SS1-SS4.

The proof makes use of the derived rule Chain. For every n, $0 \leq n < N$, we can deduce the following three formulas:

1. $\Box[\zeta_n \wedge \Delta \neq \alpha_{n+1} \Rightarrow \bigcirc\zeta_n]$
   For $0 \leq i < n$, m_i is disabled while ζ_n holds, and the truth of ζ_n is preserved by moves made by $m_{n+2},\ldots,m_N$. Hence, if m_{n+1} does not make a move, ζ_n remains true.
2. $\Box[\zeta_n \wedge \Delta = \alpha_{n+1} \Rightarrow \bigcirc\zeta_{n+1}]$
   If ζ_n is true and m_{n+1} moves, ζ_{n+1} will become true.
3. $\Box\Diamond\Delta = \alpha_{n+1}$
   By Lemma 2, m_{n+1} (like all other machines) makes moves infinitely often.

   Now, by Chain, we can deduce

$$\Box[\bigvee_{n=0}^{N-1} \zeta_n \Rightarrow \Diamond\zeta_N],$$

from which the formula of Lemma 3 follows. Note that this is a simple inductive proof: if ζ_n is ever true, then ζ_{n+1} will eventually be true, hence if ζ_0 ever becomes true, so will ζ_N.

We now need only deduce $\Diamond\zeta_0$ -- in other words, show that eventually $x_1,\ldots,x_N$ will all be different from x_0. We do so by proving that if there exists some value k, $0 \le k < K$, such that $x_1,\ldots,x_N$ are all different from k, then eventually x_0 and only x_0 will be equal to k, meaning that ζ_0 will be true.

Definition

For any l,k, $0 \le l,k < K$, let $\psi_{l,k}$ stand for

$$x_0 = l \wedge \bigwedge_{j=1}^{N} x_j \ne k.$$

By mathematical reasoning, it suffices to deduce

$$\Diamond \bigvee_{k=0}^{K-1} \psi_{k,k},$$

which means that eventually x_0 will equal some k, $0 \le k < K$, and $x_1,\ldots,x_n$ will all be different from k.

Lemma 4

For any k, $0 \le k < K$, the formula

$$\Box[\bigvee_{\substack{j=0\\ j \ne k}}^{K-1} \psi_{j,k} \Rightarrow \Diamond\psi_{k,k}]$$

can be deduced from SS1-SS4.

To prove Lemma 4, we again make use of Chain. For any j,k, $0 \le j,k < K$, $j \ne k$, we can deduce the following two formulas:

1. $\Box[\psi_{j,k} \wedge \Delta \ne \alpha_0 \Rightarrow \bigcirc\psi_{j,k}]$
   (If m_0 doesn't move, then x_0 will keep its value, namely j. Furthermore, if none of $x_i,\ldots,x_N$ has the value k, then none of them can take on that value, since any m_i, $0 < i \le N$, can only move to the state occupied by m_{i-1}.)
2. $\Box[\psi_{j,k} \wedge \Delta \ne \alpha_0 \Rightarrow \bigcirc\psi_{(j+1) \bmod K, k}]$
   (Moves made by m_0 increment its state by 1 mod K, and leave the states of all other machines unchanged.)

   By Lemma 2, we also have
3. $\Box\Diamond\,\Delta = \alpha_0$.

   The formula of Lemma 4 follows from the above three by Chain.

At this point, all that remains to be deduced is $\Diamond \bigvee_{j,k=0}^{K-1} \psi_{j,k}$. That is, we need only show that, eventually, some state k, $0 \le k < K$, will be unoccupied by all of $m_1,\ldots,m_N$. This is where we need the fact that $K \ge N$. (Incidentally, it is not difficult to see that this condition is indeed necessary.) For the case $K > N$, the proof is simple, and we omit it.

Definition

Let $\xi_{n,k}$ stand for the formula

$$x_n = k \wedge \bigwedge_{\substack{m=1\\ m \ne n}}^{N} x_m \ne k;$$

this says that m_n is the only one of the machines $m_1,...,m_N$ in state k.

We have, assuming $K=N$,

$$\Box[(\neg \bigvee_{j,k=0}^{K-1} \psi_{j,k}) \Leftrightarrow (\bigwedge_{n=1}^{N} \bigwedge_{k=0}^{K-1} \xi_{n,k})],$$

from Lemma 1, by mathematical reasoning. This means that each state k, $0 \le k < K$, is occupied by one of $m_1,...,m_N$ if and only if $m_1,...,m_N$ all occupy different states. Thus, it now suffices to prove

Lemma 5

If $K=N$, the formula

$$\Diamond \neg \bigwedge_{n=1}^{N} \bigvee_{k=0}^{K-1} \xi_{n,k}$$

can be deduced from SS1-SS4.

To prove Lemma 5, we start by deducing the following two formulas from SS1-SS4:

1. $\Box[\bigwedge_{n=1}^{N} \bigvee_{k=0}^{K=1} \xi_{n,k} \wedge \neg \bigvee_{l=1}^{N} \Delta=\alpha_l \Rightarrow \bigcirc \bigwedge_{n=1}^{N} \bigvee_{k}^{K-1}\!= 0\ \xi_{n,k}]$
   (Moves made by m_0 do not affect $x_1, \ldots, x_N$.)
2. $\Box[\bigwedge_{n=1}^{N} \bigvee_{k=0}^{K-1} \xi_{n,k} \wedge \bigvee_{l=1}^{N} \Delta=\alpha_l \Rightarrow \bigcirc \neg \bigwedge_{n=1}^{N} \bigvee_{k=0}^{K-1} \xi_{n,k}]$
   (If each state is occupied by one and only one of $m_0,...,m_N$, then a move by any one of these machines will leave some state unoccupied.)

In addition, we have, by Lemma 2,

3. $\Diamond \bigvee_{l=1}^{N} \Delta = \alpha_l$.

Now, by RUI, we can deduce from 1-3

4. $\bigwedge_{n=1}^{N} \bigvee_{k=0}^{K-1} \xi_{n,k} \Rightarrow \Diamond \neg \bigwedge_{n=1}^{N} \bigvee_{k=0}^{K-1} \xi_{n,k}$.

And finally, from 4 we conclude

5. $\Diamond \neg \bigwedge_{n=1}^{N} \bigvee_{k=0}^{K-1} \xi_{n,k}$.

This proves Lemma 5, and completes the verification.

4. Conclusion

Temporal logic is a simple but expressive language for the specification of discrete event systems. Ramadge and Wonham use regular grammars for this purpose [10], but temporal logic allows specifications to be written more naturally, and Wolper has shown how regular grammars can be combined with temporal logic in order to express properties that the temporal operators cannot [11].

The formalism also allows simple proofs. Most of the temporal reasoning in our example was captured by our simple derived rules, DCI, RUI and Chain.

Thus, temporal logic promises to be useful in control theory. Indeed, Fusaoka, Seki and Takahashi [7] have already applied tableau methods -- like those used in the synthesis of concurrent programs [5,6] -- to control problems. The synthesis of controllers from temporal logic specifications, perhaps following the approach of [7], is a topic for research. Of particular interest are synthesis procedures of a modular nature [12,13].

5. References

[1] Z. Manna and A. Pnueli, 1983. Verification of concurrent programs: a temporal proof system. Rpt. No. STAN-CS-83-967, Computer Science Dept., Stanford University, June.

[2] B.T. Hailpern, 1982. *Verifying Concurrent Processes Using Temporal Logic*. Lecture Notes in Computer Science **129**, Springer-Verlag, New York.

[3] B.T. Hailpern and S.S. Owicki, 1983. Modular verification of computer communication protocols. IEEE Trans. Commun. **COM−31**, pp. 56-58.

[4] F. Maruyama and M. Fujita, 1985. Hardware verification. Computer 18(2), pp. 22-32.

[5] Z. Manna and P. Wolper, 1981. Synthesis of communicating processes from temporal logic specifications. Rpt. No. STAN-CS-81-872, Computer Science Dept., Stanford University, September.

[6] E.A. Emerson and E.M. Clarke, 1982. Using branching time temporal logic to synthesize synchronization skeletons. Science of Computer Programming 2, pp. 241-266.

[7] A. Fusaoka, H. Seki and K. Takahashi, 1983. A decription and reasoning of plant controllers in temporal logic. Proc. 8th International Joint Conference on Artificial Intelligence, August; pp. 405-408.

[8] J.G. Thistle and W.M. Wonham, 1985. Control problems in a temporal logic framework. Systems Control Group Rpt. #8510, Dept. of Electl. Engrg., Univ. of Toronto, July.

[9] E.W. Dijkstra, 1974. Self-stabilizing systems in spite of distributed control. Comm. ACM 17, pp. 643-644.

[10] P.J. Ramadge and W.M. Wonham, 1983. Supervisory control of a class of discrete event processes. Systems Control Group Rpt. #8311, Dept. of Electl. Engrg., Univ. of Toronto, October.

[11] P. Wolper, 1981. Temporal logic can be more expressive. Proc. 22nd IEEE Symp. on Foundations of Computer Science, October; pp. 340-348.

[12] W.M. Wonham and P.J. Ramadge, 1983. On modular synthesis of supervisory control for discrete event processes. Systems Control Group Rpt. #8314, Dept. of Electl. Engrg., Univ. of Toronto, December.

[13] W.M. Wonham, 1985. On control of discrete event systems. Systems Control Group Rpt. #8508, Dept. of Electl. Engrg., Univ. of Toronto, July.

A LARGE COMMERCIAL MULTI MICROPROCESSOR SYSTEM

KENICHI UEDA AND JUN SUGANO
Matsushita Research Institute Tokyo, Inc.
Kawasaki, Japan

ABSTRACT

This paper describes the architecture of a new commercially-available large multi microprocessor system. The essential feature which distinguishes this system from the other multiprocessor systems is the realization of a very high reliability attained due to the following:

1) The implementation of dual, duplexed, and N + 1 structures according to the importance of each processor's function;

2) Controlling the system load by measuring the execution time of the particular processes at regular time intervals;

3) Automatic message sending to the substitute processor when the destination processor is discovered functionally abnormal, as evidenced in the processor state table in each processor.

INTRODUCTION

As the microprocessors are becoming wide-spread, large systems which used to be constructed of large or mini computers are now being more economically constructed using multi microprocessor systems, resulting in more compact hardware. Namely, the large systems which used to be limited to a few computer manufacturers are now manufactured by many companies. For this reason, we have developed a multi microprocessor system which provides the functions of a large computer system called MATS system (MATSushita multi microprocessor system).

The MATS system is composed mainly of INTEL's 8086 and ZILOG's z -80 microprocessors and built in a two - level hierarchical structure. There are about 70 upper level processors and about 230 lower level processors (totalling about 300) in the MATS system.

This system is a loosely coupled multiprocessor system which can easily be converted to a distributed system (especially a local network system).

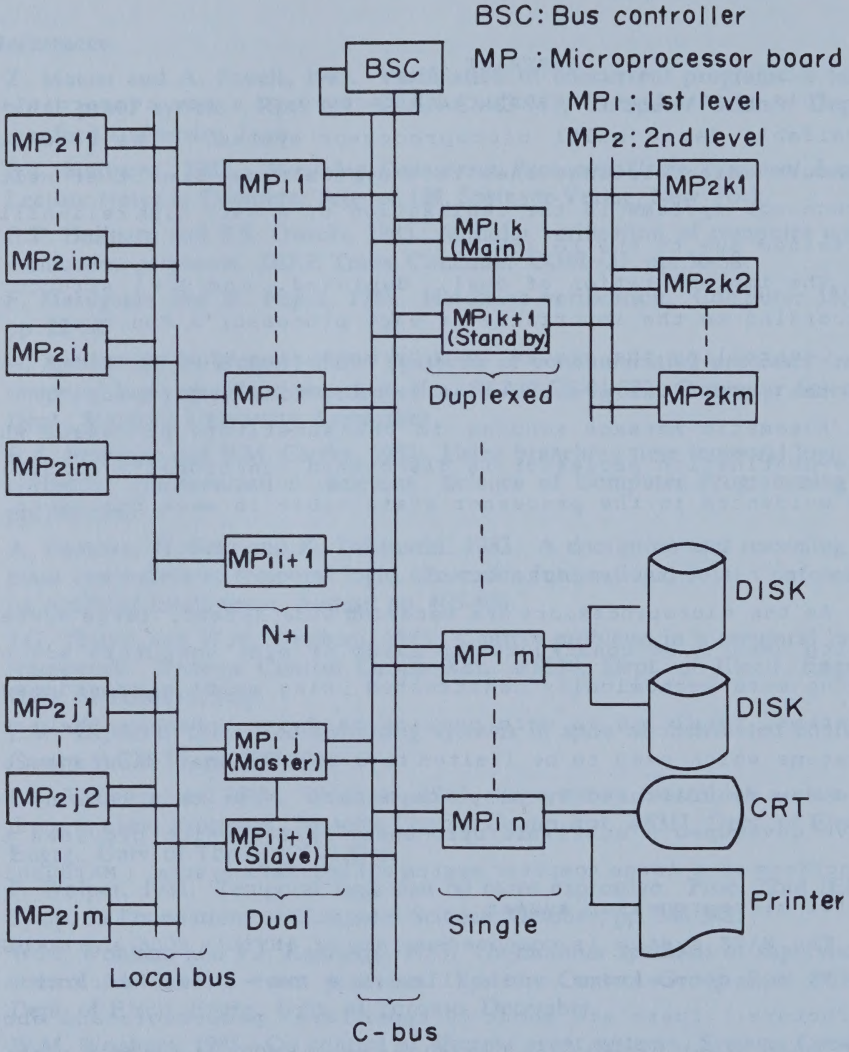

BSC: Bus controller
MP : Microprocessor board
MP1 : 1st level
MP2 : 2nd level
BSC
MP2 11
MP2 im
MP2 i1
MP2im
MP1 1
MP1 i
MP1 i+1
N+1
MP1 k (Main)
MP1 k+1 (Stand by)
Duplexed
MP2k1
MP2k2
MP2km
MP1 m
MP1 n
Single
DISK
DISK
CRT
Printer
MP2 j1
MP2 j2
MP2 jm
MP1 j (Master)
MP1 j+1 (Slave)
Dual
Local bus
C-bus

Fig. 1 The basic structure of MATS

SYSTEM CONFIGURATION

The bacic structure of the MATS system is shown in Fig.1. It is loosely coupled in a two-level hierarchical structure. Upper level processors are connected through the C-bus (a double communication bus). 8-bit parallel data is transferred through the C-bus under the control of bus-controllers. Each upper level processor controls the lower level processors or Input/Output devices directly. Maximum 32 lower level processors are connected to and controlled by one upper level processor via the communication bus called the local bus.

As mentioned above, there are two communication buses for upper level processors. Using these two buses, the load of each bus is half of conventional one-bus systems. Each processor can receive two messages from these two buses simultaneously or transmit a message using one bus and receive a message from the other at the same time.

The system employs the functional sharing and the load sharing methods according to the size of each function (Fig.2). There is no supervisory microprocessor in the upper level processor group to control all the other microprocessors in the MATS system. Each microprocessor in the upper level hierarchy performs its particular task only, whereas the processors in the lower level processor group are controlled by the upper level processors.

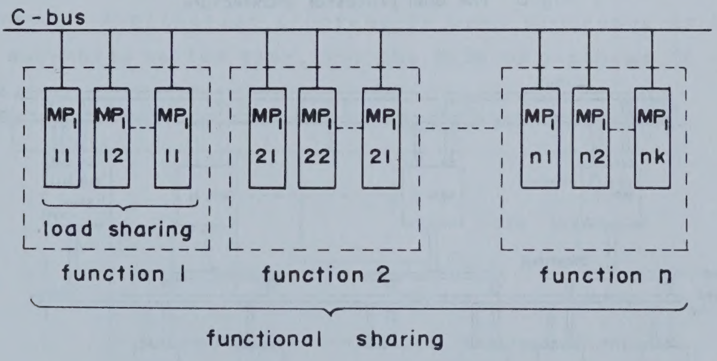


Fig.2 Functional sharing and load sharing method in the MATS system

The high degree of reliability of the MATS system is realized due to the following design concepts. The processors which carry out the functional sharing are classified in three levels in accordance with the importance of their functions:

1) The processors which play a very important role are implemented plurally so that malfunctioning of one does not shut down the entire system(Fig.3). 2) Other processors which play important roles are implemented in duplex. 3) The load sharing N-processors are implemented in the N+1 structure, where only one spare processor is allowed for N processors (Fig.4). This is how a very balanced and reliable system is realized.

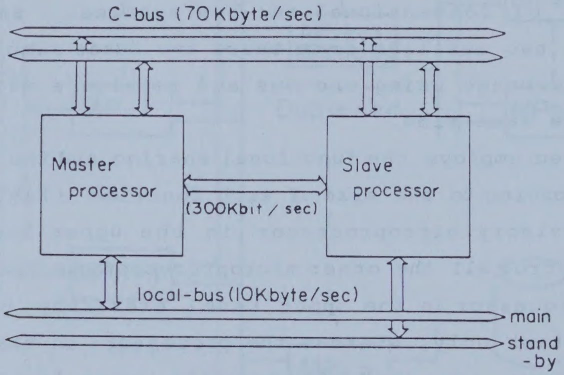

C-bus (70Kbyte/sec)
Master processor
(300Kbit / sec)
Slave processor
local-bus (10Kbyte/sec)
main
stand-by

Fig.3 The dual processor architecture

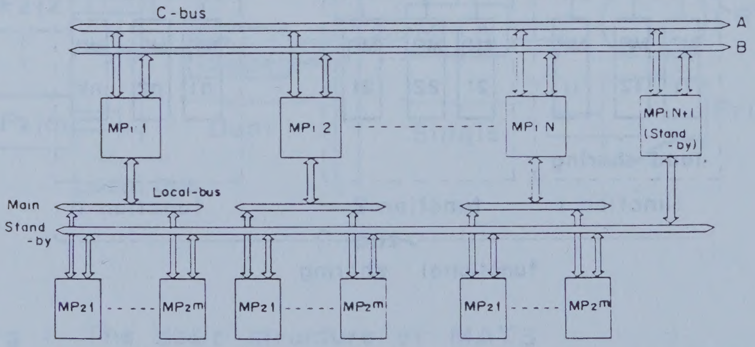

C-bus
A
B
MP1 1
MP1 2
MP1 N
MP1 N+1 (Stand-by)
Local-bus
Main
Stand-by
MP2 1
MP2 m
MP2 1
MP2 m
MP2 1
MP2 m

Fig.4 The N+1 back up configuration

Communication between upper level microprocessirs via the C-bus is carried out as follows: When a processor which is normally in the receiving mode happens to transmit the message to the other processor, it issues a bus request signal to the bus-controller, and after receiving acknowledgement from the bus-controller, it seizes the bus and transmits the message.

There are four transmission modes : broadcast, group transmission, pair transmission, and individual transmission. The pair transmission means that a packet is received simultaneously by a pair of processors in dual construction.

System software performs several types of functions, for example, file management, Input/Output device management, system maintenance, bus traffic management, and several application functions. In this system each function is assigned to one processor or more through the functional sharing method.

Software of each processor is composed of the OS, application programs, and data. Each processor has the same copy of os which is stored in read only memory.

OPERATING SYSTEM

The MATS OS provides the following functions : message transmission, lower level processor management , initial program loading, task management, processor load measurement, fault detection, error recovery, restart, and dual, duplexed, and N+1 control. Application programs in each processor is broken up into sub-units called task, and the MATS OS performs 16 - level

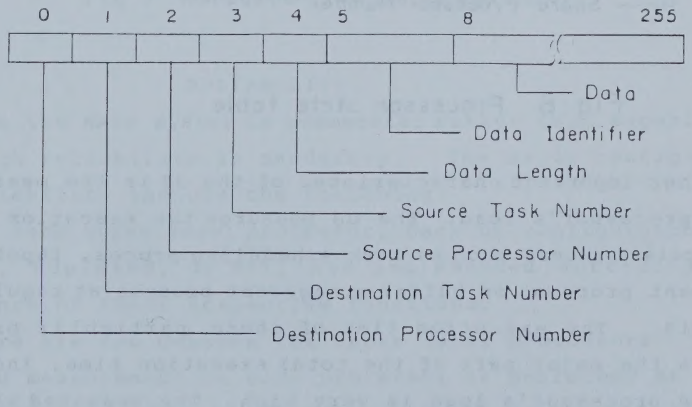


Fig. 5 Structure of C-bus packet

priority control for these tasks. These tasks are basically started up by a message via the C-bus from the other processors. That is, in a packet of C-bus message, the destination task number is included as shown in Fig.5. When the OS receives a message via the C-bus, it extracts the destination task number called the outer task number and converts it to an inner task number. It delivers the message to the task and starts it up.

One of the important characteristics of the OS's functions is the fault tolerance. When a task intends to send a message to the other processor via the C-bus, the OS looks at the processor state table where all of the upper processor's states, such as normal, abnormal, and substitute states, are recorded (Fig.6). If the destination processor is in the abnormal state, the OS attempts to find the substitute processor to send the message. If the OS fails to find the substitute processor it returns the message to the task.

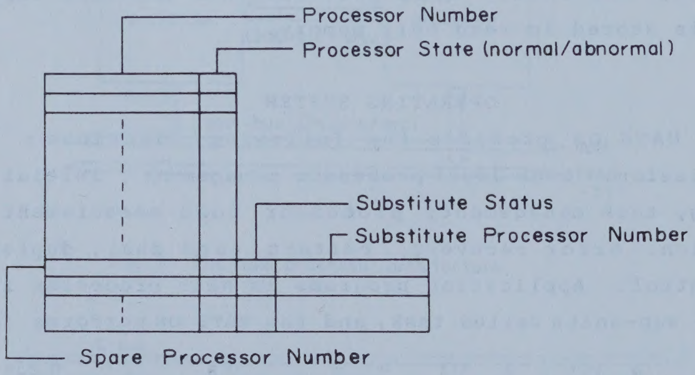

Processor Number
Processor State (normal/abnormal)
Substitute Status
Substitute Processor Number
Spare Processor Number

Fig.6 Processor state table

Another important characteristec of the OS is the measurement of the processor's load. The OS measures the execution time of a particular process such as task scheduling proces, Input/Output management process, or buffer management process at regular time intervals. The execution time of these particular processes occupies the major part of the total execution time, indicating that the processor's load is very high. The measured values of all the upper level processor's load are sent to the traffic management processor. This processor calculates the system load according to a formula using these measured values. When the load exceeds the maximum standard limit, as shown in Fig.7, the traffic management processor sends an input restriction message to all other processors. Each processor receiving the message exercises its control over the lower level processors and restricts their input data. As a result the system load tends to decrease. If the system load falls below the minimum standard limit, also shown in Fig . 7, the traffic management processor sends a message to each processor informing them to lift the input restriction. In this way, the OS is armed with a method to prevent the system from malfunctioning due to overload, and to effect smoother operation of the system.

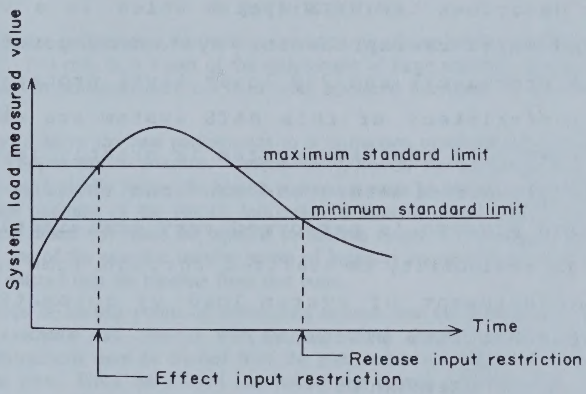


Fig.7 Management of the system load

RELIABILITY

Since the MATS system is commercial rather than experimental, very hegh reliability is mandatory. The major configurations for reliability include the following:

(1) For each upper level processor, back up configurations such as dual, duplexed, or N+1, are implemented according to the importance of their respective functions.

(2) There are two C-buses for upper level processors.

(3) Load measurement on each processor is performed at regular time intervals. The traffic management processor calculates the system load using each processor's part. It directs each

processor to restrict its input if that input exceeds the standard amount, thus preventing the system from malfunctioning.

(4) Each processor has a processor state table in which all the upper level processor's state are recorded. When a message transmission is to occur, the OS uses this table to check the status of the destination processor to see if it is faulty. If this is the case, it sends the message to the substitute processor automatically.

(5) Each processor has several diagnostic or test program. If abnormal conditions are detected due to these program executions, the OS isolates the processor from the C-buses to prevent the faulty processor from degrading the system.

SUMMARY

This paper describes the MATS system which is a very large loosely coupled multi-microprocessor system composed of about 70 upper level processors and 230 lower level processors. The important characteristecs of this MATS system are as follows:

(1) The performance of the MATS system is greatly improved due to the functional sharing method and the load sharing method by which a pipeline process is performed very efficiently.

(2) A very high reliabillty is realized through some new ideas, such as the measurement of system load or automatic message sending to the substitute processor.

REFERENCES

(1) S.H.Fuller et al., "Multi-Microprocessors: An Overview and Working Example", Proc.IEEE, Vol.66, No.2, Feb.1978, pp.216-228.

(2) R. Kober, "The Multiprocessor System SMS 201-Combining 128 Microprocessors to a Powerful Computer", COMPCON Fall 77, Washigton, D.C. Sept. 1977, pp. 225-230.

(3) Daniel D. Gajski and Jih-Kwon Peir, "Essential Issues in Multiprocessor System", COMPUTER, Vol.18, No.6, June 1985, pp. 9-27.

(4) Peter C.Patton, "Multiprocessors : Architecture and Applications", COMPUTER, Vol.18, No.6, June 1985, pp.29-40.

LOW COST BRANCH PREDICTION

David A. Poplawski[†]
Department of Mathematical and Computer Sciences
Michigan Technological University
Houghton, Michigan 49931

ABSTRACT

Most branch instructions in programs behave very predictably, that is, a very high percentage either branch or don't branch a very high percentage of the time, and the predictability is consistent over a range of input data for each given program. This property is used as the basis for a branch prediction strategy that relies on an execution signature of a program to make a branch/fall through prediction for each branch. The signature is obtained by direct measurement of the behavior of each branch instruction during a small number of characteristic executions of a given program. The prediction accuracy of this scheme compares favorably with expensive branch prediction strategies, and can be achieved with no hardware support other than a doubling of the number of conditional branch opcodes.

1. INTRODUCTION.

To attain better performance, instruction execution pipelining has become prevalent in many computers today [2]. Not only is it a part of the architecture of large mainframes and supercomputers, but can also be found in minicomputers and even more advanced microprocessors (in the form of instruction prefetching).

In order to achieve the best performance in a instruction pipelined machine, it is important that the pipeline be kept as full as possible. Unfortunately, branch instructions present a difficulty in that the instructions that logically follow the branch instruction must be fetched and entered into the pipeline **before** the actual outcome of the branch instruction is known. Since waiting for the outcome of a branch to be determined will cause the pipeline to become empty, a *branch prediction strategy* is often used to choose one of the possible starting points of logically succeeding instructions, and instructions are fetched and entered into the pipeline from that point.

If the choice of starting points of instructions to enter into the pipeline is eventually determined to be correct, execution can continue with the instructions already in the pipe; if the choice was incorrect, the instructions must be flushed from the pipe and a new sequence fetched and entered into the front of the pipe. Since the reward for making the correct prediction can be high (as can the penalty for a wrong prediction), it is an advantage to use a branch prediction strategy that is most accurate relative to the expense of implementation.

In this paper we present a branch prediction strategy that has accuracies comparable to more expensive strategies with the cost of the addition of a few branch instructions to the instruction set of the machine. A prediction for a given branch is simply based on the observed behavior of that branch in some number of previous executions of the program. If the branch has been observed to branch more often than not, it will be predicted to branch in all future executions, otherwise it will be predicted to not branch. This simple strategy leads to remarkably good prediction accuracies.

2. PREDICTION STRATEGIES.

Branch prediction strategies have be divided into two general classes - static and dynamic [3]. Static strategies, which are the least expensive to implement, are based on intelligent guesses as to the execution behavior of classes of branch instructions and are fixed in the architecture of the machine.

[†] This work partially supported by NSF Grant DCR-8404909.

There are many types of static strategies, including always predicting that a branch will branch, predicting that certain on branch op-codes will branch and others will not, predicting that a branch forward in the program will not branch and a branch backwards will, etc. Unfortunately, static branch prediction strategies do not, in general, produce very good prediction accuracies. For example, in [1], Lee and Smith report average prediction accuracies of between 60 and 70 percent for various static strategies. Smith [3] gives a range of 65.7 to 99.4 for various strategies on various programs, with a mean of 86.7. Our data averages approximately 79 percent for the best static strategy (based on opcode).

Dynamic strategies use the behavior of an executing program to make branch predictions. For example, predicting that a given branch instruction in a program will branch (not branch) if it branched (did not branch) the last time it was executed, is a dynamic strategy. In general, dynamic strategies have better prediction accuracies than static. Smith [3] reports accuracies above 90 percent when a small amount of history about **every** branch is used - realistically the history of only a small subset of the branches can be retained and therefore the prediction accuracy is reduced. However, dynamic strategies usually require considerable hardware and are therefore much more expensive to implement. For example, the Branch Target Buffer [1] consists of a high-speed cache memory with anywhere from 64 to 256 entries, each of which contains two addresses and a few bits of state information to record the behavior of the most recently executed branch instructions.

In contrast to either static or dynamic strategies, ours relies on an execution *signature* of the program to make a branch/fall through prediction for each branch. The signature is a record of the branching behavior of each unique branch instruction in the program. It is obtained by direct measurement of the behavior of each branch instruction during one or more *characteristic executions* of the program on varying data, and is reduced to two values for each branch: the number of times it branched and the number of times it didn't. In further executions, each branch will be predicted to branch if the number of times is branched in the signature exceeded the number of times it didn't, and it will be predicted to not branch if the number of times is branched in the signature did not exceed the number of times it didn't.

Signatures are obtained by a modification the compiler that cause it to insert extra recording instructions into the object code for the program to be measured and then executing the program on typical data and dumping the branch behavior to a file after execution. Once obtained, a signature (or the combination of several from different executions on different data) is used to modify the object program to indicate, for each branch, to predict whether it will branch or not.

3. BRANCH BEHAVIOR.

The C and FORTRAN compilers on Berkeley UNIX were modified to generate extra code that would count, for each branch instruction in a program, the number of times it branched and the number of times it didn't. The actual branch behavior resulting from an execution of the program on a VAX 11/750 was then dumped to a file upon completion of the program. Files from various executions of programs were then post-processed to obtain the results reported in this paper.[1]

Several C and FORTRAN programs were selected to test the performance of the branch prediction method. The C programs were the standard UNIX commands du, ls, and spell, the VAX portable C compiler, a gauss elimination program, and an instruction level multiprocessor simulator. The FORTRAN programs were a gauss elimination program (translated from the C version) and a matrix inversion program. These programs amounted to almost 20,000 lines of code. The branch behavior and prediction data is based on over 124 million branch executions of these 8 programs on 162 different data sets.

The data shows that the outcome of each branch instruction in a given program when run with a given input data set is very predictable. Table I shows the percentage of statically occurring branches that have a given predictability (i.e., prediction accuracy) for two programs: the first is an execution of the portable C compiler compiling one of its own component source files (code.c), and the other is an execution of the gauss elimination program on 47 variables.

As can be seen, 78.8 percent of the branches in the compiler and 79.6 percent of the branches in the gauss elimination program were completely (100 percent) predictable - that is, they either branched every time they were executed or they did not branch every time. This includes both unconditional branches (which always branch) and conditional branch instructions. Furthermore, 86.9 percent of all branches in the compiler were greater than 90 percent predictable - that is, each of the 86.9 percent of the branches either branched more than 90 percent of the time or did not branch more than 90 percent of the time.

---

[1] All results are based on both unconditional and conditional branch instructions so they can be compared with other reported results, which do the same.

The results above do not take into account the frequency of execution of the branch instructions, and therefore can not be used to estimate prediction accuracies. Actual predictability is based on the total number of executions of each branch (a dynamic measure) rather than just its appearance in the executable object file of a program. Table II shows the results when the branch results are weighted by frequency of execution.

From this table we see that of all branches executed, 51 percent of them in the compiler and 55.9 percent of them in the gauss elimination program were 100 percent predictable, and 83.3 percent in the compiler and 98 percent in the gauss elimination program were at least 90 percent predictable.

Data for other programs measured show very similar properties - that is, a high percentage of all executed branch instructions are very predictable. Table III shows the percentage of instructions executed that were at least 90 percent predictable for the other programs measured.

These results indicate that branches behave very predictably on any single data set. To be of use in branch prediction, it is important that the branches behave the same way on different data sets, since a signature derived from one data set is used to predict what the branches will do on another data set. To show that they do behave nicely, a single execution on an arbitrarily chosen data set was used to obtain execution signatures for each of the benchmark programs (the compiler was run on several data sets and the signatures combined into one). Then the prediction accuracy for each test program was measured several times with several arbitrarily chosen data sets. The branch prediction accuracy for each program is shown in Table IV.

These prediction accuracies compare very favorably with dynamic strategies having accuracies above 90 percent. Even the worst prediction was 85.2 percent. For comparison, static prediction accuracies averaging 76 percent for predict branch always and 79 percent for predict based on opcode (using optimally chosen predictions for each opcode) were obtained on the same data.

4. IMPLEMENTATION.

A simple extension to the instruction set of the machine provides the actual mechanism for doing branch prediction. Each conditional branch instruction in the instruction set is replaced by two instructions - one which predicts the branch is taken and one which predicts it is not. For example, the BNEQ (branch not equal) instruction in the VAX instruction set would be replaced by two instructions, BNEQ-taken and BNEQ-nottaken. When the instruction is fetched, the opcode is used to make the prediction - BNEQ-taken predicts that the branch instruction will branch when executed, BNEQ-nottaken predicts that it will not. Each branch that would normally appear in a program is replaced by one of the predictive branch instructions - the one chosen is based on the execution signature.

The hardware cost of this mechanism is minimal. In a microcoded machine, there may be no cost at all unless the extra microinstructions do not fit into the available microstore, in which case extra space would be required. If the machine already has a static branch prediction then the cost would be no greater - perhaps less, depending on the complexity of the static strategy. The cost is significantly less than any of the dynamic strategies, which require various mechanisms for recording the recent behavior of branch instruction during execution.

This implementation will also support any other static branch prediction strategy. All that is necessary is that the compiler generate the appropriate predicting branch in each case. For example, to implement a branch always strategy, the compiler would generate branch-taken in every case. To implement predict backward taken, forward not taken, the compiler would generate branch-taken branches for all backward branches and branch-nottaken for all forward ones. This feature can be used for getting reasonable prediction accuracies when no signature exists. Such might be the case for programs that are run only a very few times, or for branches in a program that has a signature which indicates the branch wasn't executed at all.

5. SIGNATURE GATHERING.

The simplest way to obtain the execution signature of a program is to modify the compiler to generate extra "recording" instructions into the object code to be measured. This technique requires no hardware support and can be enabled by a simple parameter to the compiler. It has the disadvantage of increasing the execution time of the program being measured by a fairly significant amount (the test programs were slowed by 30-40 percent). A more expensive, but considerably faster, method is to have hardware that will record the behavior of each branch instruction. Because the primary advantage of this branch prediction strategy is the minimization of hardware cost, this method will not be elaborated.

Even though there is an overhead with the software measurement technique, the modified programs need not be executed very often before a signature that will produce good branch prediction accuracies is obtained. Consequently the overall measurement overhead is reasonable. All but one of the prediction accuracies reported in this paper were attained with one characteristic execution each program, and the data chosen for that execution was arbitrarily chosen. Even when several characteristic executions were made (in the case of the compiler), prediction accuracies were not significantly better that when just one of the executions was chosen for the signature.

A single characteristic execution on arbitrarily chosen data has been shown, in the cases tested, to produce good prediction accuracies. It is not known that this will be true in general without examining more cases. One potential problem is that many branch instructions were not executed at all during the characteristic execution(s). Often as many as 20-30 percent of the branches were not executed. Consequently there is nothing to base a prediction on. One can use an existing static prediction strategy for predicting those branches (this is in fact what was done - otherwise unpredicted branches were predicted to be taken). It has fortunately been the case that few of the branches for which no signature-based prediction could be made were executed when the prediction accuracy was being measured This is probably attributable to the fact that most of the unexecuted branches were in infrequently executed parts of the code such as error handling, and doing poor branch prediction in infrequently executed code doesn't cost much.

Since few characteristic executions were necessary to get good prediction accuracies, one might wonder when enough runs have been made to maximize the prediction accuracy. The following method seems to work well. First, make two characteristic executions. Count the number of branches whose predictions would be different between the first signature and the merged signatures of the two characteristic executions. If the count is zero, then no new information was provided by the second execution, and perhaps no more characteristic executions are needed. One cannot necessarily infer that further characteristic executions will also provide no information, but so far one execution would have been enough.

It is unlikely that the second execution provides no new information. In this case, make another characteristic execution, and count the number of branches whose predictions would be different between the merged signatures of the first two executions and the merged signature of all three executions (notice that these merged signatures will be more "typical" since the branch behavior for several data sets will have been combined). If this new count is zero, then again no new information has been provided and there may be an argument for using the existing signature to make predictions.

The process of making another characteristic execution and inspecting the count of changed predictions can be continued until the count remains zero several times. This provides confidence that further runs will also provide no information. However, it may be the case that there are always some changes. This would be true for programs that have one or more branches that behave very close to 50-50. One execution may give a ratio of 49-51 and the next a ratio of 52-48, thereby causing the predictions for those branches to change. Succeeding executions might only toggle this "unpredictable" branch back and forth, and the count of changed predictions will never become zero. In this situation the sequence of changed prediction counts over several merged signatures should eventually stabilize near some value. When this happens, enough characteristic executions have probably been made.

Finding "representative" data sets for a given program can be a problem. For some programs almost reasonably sized data set is representative - such programs are relatively insensitive to their data (many numerical programs have this property). Other programs are more sensitive, and more data sets will be required to get good predictions (on the average). A good check of the sensitivity of a program is the number of characteristic executions are required to procedure a stabilized count of changed predictions as outlined above.

One obvious source of representative data is the data used to debug and test the program. Debugging data has the advantage of being designed to exercise all parts of the program, and so there should be branch behavior data for all branches. Acceptance and shakedown testing data should be designed to be somewhat typical in the first place.

6. CONCLUSIONS.

The observed branching behavior of a few executions of a program provides enough information to drive a low cost but highly accurate branch prediction mechanism. While not directly applicable to programs that are run only once, most programs do not fall into that category. Even for programs that are run only once, the compiler could in principle generate a predicting branch instruction according to some reasonably accurate static branch prediction scheme and thereby improve upon the accuracy of a totally random strategy - the minimal hardware support required for signature-based branch prediction will also support any static strategy.

The fact that only a few representative executions of a program are enough to produce excellent prediction accuracies is a great benefit of the strategy. Production quality programs, which are responsible for most of the execution time on most computers (e.g., operating systems, compilers, word and text processors, etc.) can easily be measured during the final testing stages, where representative data that exercises all parts of the program is already being used. As a result, this branch prediction strategy can result in very high prediction accuracies (and therefore shorter program execution times) with virtually no hardware cost and with minimal overhead that can be absorbed into normal testing stages.

7. REFERENCES.

1. J. K. F. Lee and A. J. Smith, Branch prediction strategies and branch target buffer design, *Computer 17*,1 (Jan 1984), 6-22.
2. C. V. Ramamoorthy and H. F. Li, Pipeline architecture, *Computing Surveys 9*,1 (Mar. 1977), 61-102.
3. J. E. Smith, A study of branch prediction strategies, *Proceedings, 8th Annual Symposium on Computer Architecture*, 1981.

Compiler			Gauss Elimination		
Prediction Accuracy	% of Branches	Cumulative % of Branches	Prediction Accuracy	% of Branches	Cumulative % of Branches
100%	78.8%	78.8%	100%	79.6%	79.6%
>=99%	2.5%	81.3%	>=99%	2.0%	81.6%
>=98%	1.2%	82.5%	>=98%	8.2%	89.8%
>=97%	0.7%	83.2%	>=97%	0.0%	89.8%
>=96%	1.0%	84.1%	>=96%	4.1%	93.9%
>=95%	0.4%	84.5%	>=95%	2.0%	95.9%
>=94%	0.6%	85.1%	>=94%	0.0%	95.9%
>=93%	0.3%	85.4%	>=93%	0.0%	95.9%
>=92%	0.5%	85.9%	>=92%	0.0%	95.9%
>=91%	0.5%	86.4%	>=91%	2.0%	98.0%
>=90%	0.4%	86.9%	>=90%	0.0%	98.0%
>=80%	4.4%	91.2%	>=80%	0.0%	98.0%
>=70%	2.8%	94.0%	>=70%	0.0%	98.0%
>=60%	2.7%	96.7%	>=60%	2.0%	100.0%
>=50%	2.4%	99.2%	>=50%	0.0%	100.0%
>=0%	0.8%	100.0%	>=0%	0.0%	100.0%

Table I. Predictability of Statically Occurring Branches.

Compiler			Gauss elimination		
Prediction Accuracy	% of Branches	Cumulative % of Branches	Prediction Accuracy	% of Branches	Cumulative % of Branches
100%	51.0%	51.0%	100%	55.9%	55.9%
>=99%	19.2%	70.2%	>=99%	3.2%	59.1%
>=98%	2.0%	72.2%	>=98%	3.0%	62.0%
>=97%	0.8%	73.0%	>=97%	0.0%	62.0%
>=96%	0.9%	73.9%	>=96%	36.5%	98.5%
>=95%	0.5%	74.3%	>=95%	0.7%	99.2%
>=94%	1.1%	75.4%	>=94%	0.0%	99.2%
>=93%	0.5%	76.0%	>=93%	0.0%	99.2%
>=92%	6.9%	82.8%	>=92%	0.0%	99.2%
>=91%	0.3%	83.1%	>=91%	0.0%	99.3%
>=90%	0.2%	83.3%	>=90%	0.0%	99.3%
>=80%	4.5%	87.9%	>=80%	0.0%	99.3%
>=70%	3.1%	91.0%	>=70%	0.0%	99.3%
>=60%	3.0%	94.0%	>=60%	0.7%	100.0%
>=50%	4.0%	98.0%	>=50%	0.0%	100.0%
>=0%	2.0%	100.0%	>=0%	0.0%	100.0%

Table II. Predictability of Dynamically Occurring Branches.

Program	>=90% Predictable
du	79.2%
ls	84.1%
spell	97.1%
compiler	83.3%
gauss elimination (C)	99.3%
simulator	89.8%
gauss elimination (F77)	98.7%
matrix inversion	64.6%

Table III. Dynamic Predictability.

	Data	Prediction Accuracy		
Program	Sets	Minimum	Average	Maximum
du	20	89.6%	96.0%	99.7%
ls	20	85.2%	90.5%	94.8%
spell	6	96.8%	97.1%	98.5%
compiler	69	89.2%	92.1%	95.6%
gauss	6	94.7%	96.9%	98.5%
sim	5	96.3%	97.1%	98.0%
gauss(f77)	6	87.1%	94.0%	98.2%
invert(f77)	30	90.1%	91.3%	91.9%

Table IV. Prediction Accuracies.

AN EFFICIENT ARCHITECTURE FOR IMPLEMENTING SPARSE ARRAY VARIABLES

JOHN T. O'DONNELL
Computer Science Department
Indiana University
Bloomington, Indiana 47405

ABSTRACT

Array variables provide a powerful set of access functions that support a robust programming methodology, but with conventional implementation techniques some of the access functions require iteration. Therefore those access functions are very slow and take unpredictable time. The Array Variable Machine is a tree structured memory system that implements all of the access functions in a constant number of cycles. In addition, it supports sparse array variables (a generalization of Dijkstra's array variables). There is a continuum from very sparse to completely dense; the AVM makes all access functions equally efficient throughout the entire continuum.

INTRODUCTION

In *A Discipline of Programming* [3, chap. 11], Edsger Dijkstra proposed the *array variable* as an alternative to the conventional notion of arrays. In the conventional view an array is not an individual object, but instead is a block of adjacent storage words that contain its *elements*, identified by consecutive integer *indices*. The computer's address calculation hardware provides for efficient access to any array element, given its index. There are, however, no operations that act directly on entire arrays.

An array variable is an individual object, and we think of the array variable operations as acting on that entire object, rather than just acting on an individual element of the object. All the operations on array variables are intended to be efficient and roughly comparable in their execution time, and they must always leave the value of the array variable completely defined. One consequence of these constraints is that there is no operation that can quickly allocate a large array, leaving all its elements undefined. Instead, there are operations that can create a new array variable with a single defined element, and there are operations that can extend the *domain* of the array variable (i.e., the set of indices for which a value is defined).

Array variables have some of the properties of conventional arrays, but they also share some of the properties of linked lists. Consequently they are hard to implement efficiently on conventional architectures. Dijkstra suggests that an implementation could solve these problems by allowing the user to state limitations on the indices, but we are concerned with efficient *general* implementations that do not require the user to provide such information.

This paper describes a computer storage architecture, called the Array Variable Machine (AVM), that can efficiently implement all the array variable operations in constant time. The architecture also provides very fast implementation of several data structure operations required by LISP and Scheme, including dynamic binding access, static binding access, and combined list/vector structures. First we describe the access functions defined on array variables and the difficulty of implementing them on a conventional architecture. Then we describe the architecture of the AVM, and show how it represents sparse array variables. The conclusion compares the AVM approach with RAM memories and with other models of parallelism.

ACCESS FUNCTIONS

An empty array variable has the value **nil**. A non-empty array variable A has a lower bound *low-bound*(A) and an upper bound *high-bound*(A) such that $A[i]$ is defined for *low-bound*(A) $\leq i \leq$ *high-bound*(A), and $A[i]$ is undefined for $i <$ *low-bound*(A) and for $i >$ *high-bound*(A). There are a number of access functions that allow the programmer to fetch information from a sparse array variable and to modify it. A few of these are:

- *store*(A, i, x)
  Sets $A[i] := x$ (this may extend the domain).
- *fetch*(A, i)
  Returns A[i] if *low-bound*(A) $\leq i \leq$ *high-bound*(A);
  indicates **error** otherwise.
- *low-bound*(A)
  Returns the lower bound of A; **false** if $A =$ **nil**.
- *high-bound*(A)
  Returns the upper bound of A; **false** if $A =$ **nil**.
- *low-extend*(A, x)
  Sets $A[$*low-bound*$(A) - 1] := x$ (extends domain); indicates an error if A=**nil**.
- *high-extend*(A, x)
  Sets $A[$*high-bound*$(A) + 1] := x$ (extends domain); indicates an error if A=**nil**.
- *low-remove*(A)
  Removes *low-bound*(A) from the domain of A.
- *high-remove*(A)
  Removes *high-bound*(A) from the domain of A.
- *next*(A, i, i')
  Sets $i' := j$, where $j > i$, $A[j] \neq 0$ and $A[k] = 0$ for $i < k < j$.
  If no such j exists, sets i to **end**.

The domain of an array variable A is $\{i \mid$ *low-bound*$(A) \leq i \leq$ *high-bound*$(A)\}$, and $A[i]$ is always defined if i is in the domain of A. Thus an array variable created with *store*(A, 0, 100) and *store*(A, 25, 200) has all its elements $A[i] = 0$ for $0 < i < 25$. The domain extension and domain element removal functions give array variables some of the properties of LISP lists, while *store* and *fetch* correspond to conventional indexing.

PROBLEMS WITH CONVENTIONAL IMPLEMENTATIONS

A useful implementation should meet several requirements. First, all of the access functions should be relatively fast. Second, there should not be much variation in the amount of time any function requires — i.e., the execution time of an access function should not depend on the size or the sparseness of the array variable. Finally, the implementation should use storage efficiently, and it should not abort a program if there is still a large amount of available storage.

Three conventional candidates [6] for representing array variables are described below: (1) sequential allocation with indexing, (2) heap allocation, and (3) hashing.

Sequential allocation (as in conventional arrays) makes the *fetch* function fast, but leads to problems with the functions that can extend the domain (*low-extend*, *high-extend* and *store*). If we represent each array variable compactly, with no adjacent

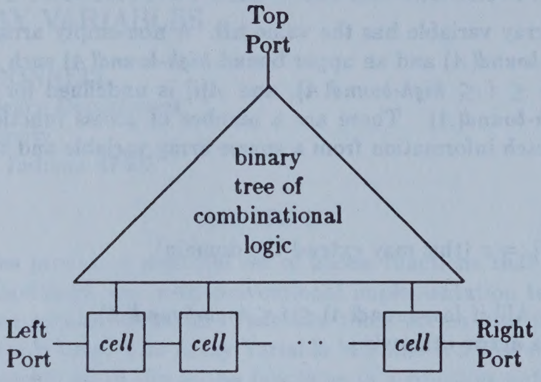


Figure 1. Architecture of the AVM Storage

empty storage, it is impossible to extend the domain without recopying one or more array variables. This violates our requirement that all access functions take a small bounded amount of time. To avoid recopying array variables, we could allocate in advance a large area for each array variable. However, this approach is too inflexible, because a domain extension of one large array variable can cause the program to fail even when other array variables do not completely fill their areas.

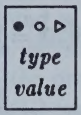
Heap (or linked list) allocation solves the storage management problems raised by the domain extensions, but the fetch and store functions would require iteration (since indexing into a scattered list is impossible). Again, this violates our requirements for the implementation.

Another approach is to store all array variable elements in a global hash table. The hash key for element $A[i]$ is formed from both A and i. This representation makes both the random access functions (*fetch*, *store*) and the domain extension functions efficient as long as the table does not become too full. However, the *next*, *low-remove* and *high-remove* functions become hard to implement, and performance degrades as storage becomes full.

All of the access functions can be implemented using conventional techniques while satisfying the constraints; the problem is that some access functions are efficient over one representation while other access functions require a different representation. The rest of this paper describes an architecture that can satisfy the constraints while implementing all of the access functions efficiently on one representation.

THE ARRAY VARIABLE MACHINE

A conventional computer storage can execute two instructions, *fetch* and *store*, which are sufficient for the implementation of ordinary data structures. In contrast, the Array Variable Machine is an active storage unit that can execute a number of instructions useful for implementing array variables. In addition to *fetch* and *store*, the AVM instructions can perform operations that would require iteration using a conventional storage. Figure 1 shows the architecture of the AVM.



select	●	indicates active cell
mark	○	marks a span of adjacent cells
attached	▷	indicates continuation of data
type		data type of value
value		data value

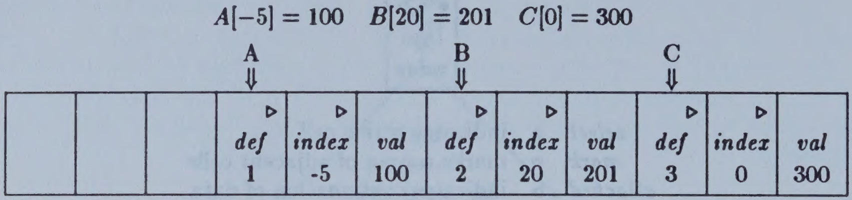
Figure 2. AVM Storage Cell Fields

The storage is organized as a linear sequence of *cells*. Instead of using addresses to refer to cells, as in the von Neumann architecture, the AVM always refers to cells through associative access [4]. We shall call one of the endpoint cells the *leftmost* and the other one the *rightmost* (we cannot refer to their addresses because the cells don't have addresses). Each cell has a unique predecessor and successor. The predecessor of the leftmost cell is the *Left Port* and the successor of the rightmost cell is the *Right Port*.

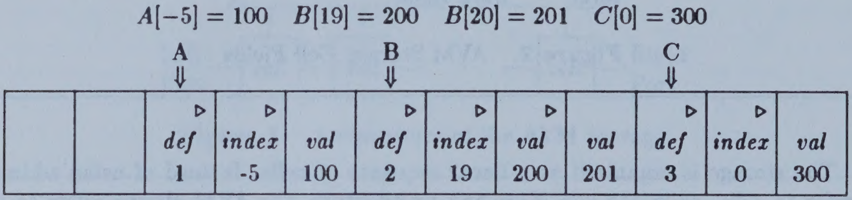
Figure 2 shows the contents of a storage cell. Each cell contains two fields (Type and Value) and three flags (*select*, *mark* and *attached*). The Type field indicates the data type of the value stored in the Value field (thus the AVM is a tagged memory). The choice of data types depends on the intended application for the AVM; we assume that at least three types exist: *def*, *index* and *val*. The *val* type indicates the value of an array variable element. The *def* type indicates that the Value field contains a unique name (i.e., no other cell with Type=*def* may have the same Value). Each array variable representation begins with a *def*, allowing associative access to the representation. The *index* type indicates that the Value field contains the index of the next element (which is of type *val*).

There are three groups of storage instructions: instructions that manipulate the *select* flag (•); instructions that manipulate the *mark* flag (○); and instructions that shift adjacent sets of words. The data paths connecting neighboring cells allow large data structures to be shifted in one cycle; the domain extension functions use shifts to insert new elements into a representation. The AVM architecture must perform associative searching [4], indexing, vector size measurement and *mark* flag computation, which all require logic operations on global information about the state of the machine. A binary tree of combinational logic performs such computations, and it also broadcasts instructions to all the storage cells. The root of the tree is a port to the processor (the Top Port), and the leaves of the tree are connected to the storage cells.

(a) initial representation


A[−5] = 100 B[20] = 201 C[0] = 300
A
B
C
def 1
index -5
val 100
def 2
index 20
val 201
def 3
index 0
val 300

(b) representation after B[19]:=200 (*"low-extend"*)


A[−5] = 100 B[19] = 200 B[20] = 201 C[0] = 300
A
B
C
def 1
index -5
val 100
def 2
index 19
val 200
val 201
def 3
index 0
val 300

(c) representation after B[21]:=202 (*"high-extend"*)

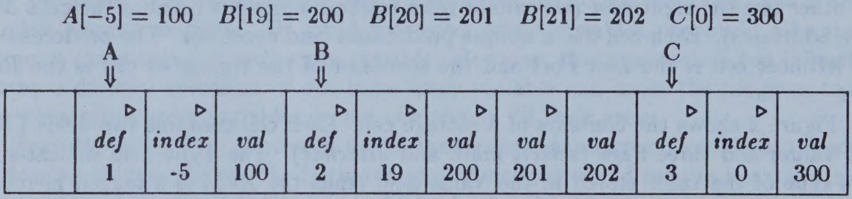

A[−5] = 100 B[19] = 200 B[20] = 201 B[21] = 202 C[0] = 300
A
B
C
def 1
index -5
val 100
def 2
index 19
val 200
val 201
val 202
def 3
index 0
val 300

Figure 3. Examples of dense array variable representations.

THE AVM DATA STRUCTURES

This section illustrates the data structures used to represent array variables on the AVM, and gives an example of the *fetch* access function which demonstrates some of the AVM instruction set.

Figure 3 illustrates the representation of dense array variables. (Sparse arrays, described below, include dense arrays as a special case.) Figure 3a represents three array variables, A, B and C. Each array variable representation begins with a word of type *def* that specifies the "name" of the array variable. For example, the names of A, B and C are 1, 2 and 3 respectively. The array variables are accessed associatively through their names; addresses are not used because the domain extension functions cause entire arrays to be moved — thus the address of an array variable is not constant.

Following the *def* is a cell with type *index* which contains the index of the subsequent *val* cell. Thus the *index* of -5 in the representation of A indicates that the following *val* "100" has index -5; i.e., $A[-5] = 100$.

The *attached* flags indicate how many cells constitute the representation of an array variable. If *attached* in a cell is set (indicated by ▷ in the figure), then the representation continues into the next cell; otherwise that cell is the last in the representation. Thus Figure 3a shows three distinct array variable representations, each filling three cells.

Figure 3b shows the result of a *low-extend* function. Note that the *index* for B has been decremented, because the lower bound is reduced by 1. The *val* "200" has index 19, and the *val* "201" has index 20. In general, if a *val* is preceded by another *val*, its index is 1 greater than the index of the preceding *val*. Figure 3c shows the result after a *high-extend*. A new *val* "202" has been inserted, and the *attached* flag in the cell containing *val* "201" has been set because the representation now continues into the cell containing "202".

Sparse array variables are represented in a similar way, except a single array variable representation may contain several *index* cells. Any array element that is not explicitly represented by a *val* is assumed to be 0. Figure 4 shows how a sparse array variable can be created, and how its representation automatically turns into a dense representation as the "zero elements" in the domain are assigned non-zero values.

In Figure 4a, the array variable A is created with one element — thus it starts out as a dense representation. All array variables must be created this way. Figure 4b shows the result of an assignment to $A[3]$. Since the previous lower bound of A was 0, this leads to a sparse representation, with the elements $A[1]$ and $A[2]$ defined as 0 but not represented explicitly. Each group of adjacent non-zero elements is represented by an *index* giving the index of the first element, followed by the element values in cells of type *val*. In the worst case (extremely sparse) it takes two cells to represent each array element: one for its index and one for its value. Figure 4c shows that an assignment to $A[1]$ leads to a representation that is partly sparse (since $A[2]$ is not represented explicitly) and that is partly dense (since $A[0]$ and $A[1]$ have consecutive indices, they need only one *index* cell). The microprogram that implements the access functions checks to see whether it can make a dense representation whenever it defines a new array element; this takes a small constant number of cycles. In Figure 4d the sparse "hole" in the array, $A[2]$, has been assigned a non-zero value. The implementation discovers (again in a small constant time) that it can replace the *index* with the new *val*, leading to a completely dense representation.

One of the chief advantages of the AVM is its efficient representation and accessing of array variables throughout the entire continuum from extremely sparse to completely dense structures. The conventional implementation methods do not have this property. The usual linked list representation of sparse vectors becomes gradually less effective as the vector becomes less sparse, and it works very poorly for completely dense vectors. Indexing is efficient for the conventional dense representation, and scanning from one non-zero element to the next is efficient for the conventional sparse representation. However, the converse operations — indexing in a linked list and scanning for non-zeros in a sequential array — may be inefficient using conventional methods, and their speeds depend on the sparseness of the array. On the AVM the indexing and scanning operations, and all the other access functions, are always efficient — regardless of the size or sparseness of the array.

Figure 5 shows how the indexing operation works on a sparse representation, and it illustrates some of the instructions of the AVM. Figure 5a shows a partially sparse representation of A; we want to perform the access function *fetch*$(A, 12)$. In 5b the "select" instruction sets the *select* flag in the *def* cell marking the beginning of the array variable (this is an associative access). Then the "mark-span" instruction sets the *mark* flags in all the cells that comprise the representation. The binary tree of combinational logic implements mark-span by performing a global analysis of the settings of the *select* and *attached* flags (the details are given in [10]). In 5c the "select-index" instruction sets *select* in all the cells with *mark* set such that the type of the cell is *index* and the

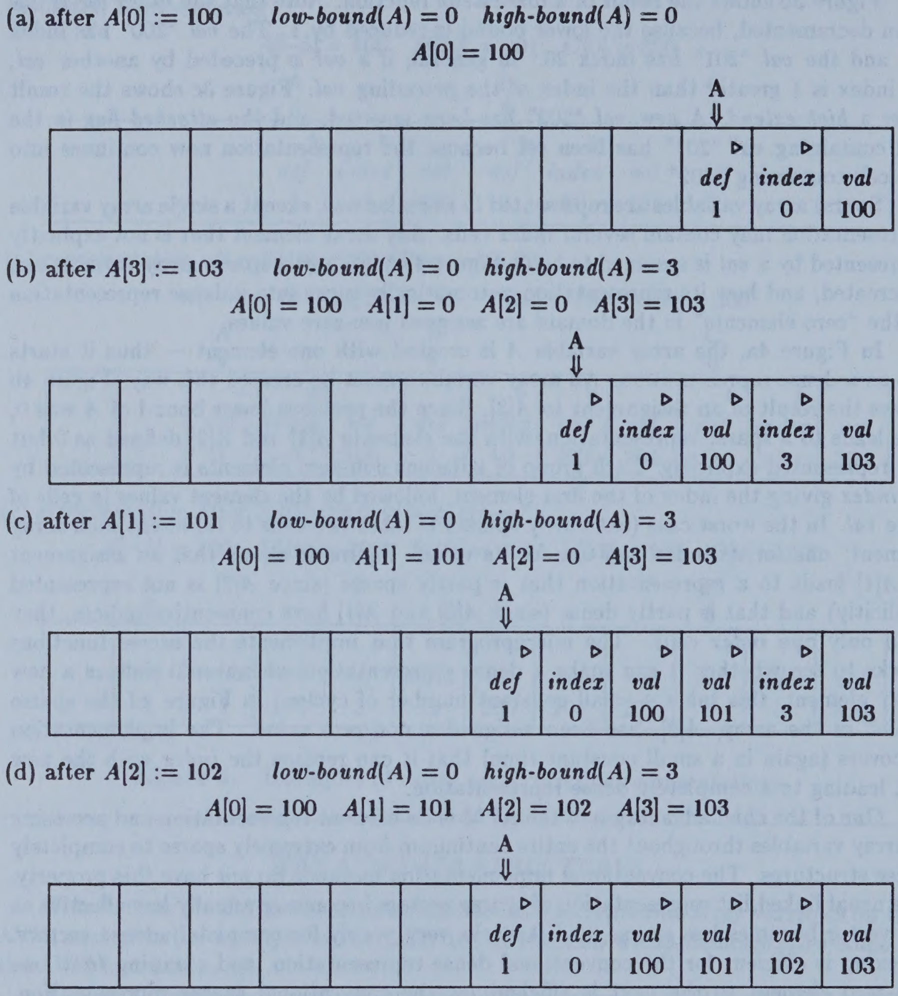

(a) after A[0] := 100 low-bound(A) = 0 high-bound(A) = 0
A[0] = 100
A
def index val
1 0 100
(b) after A[3] := 103 low-bound(A) = 0 high-bound(A) = 3
A[0] = 100 A[1] = 0 A[2] = 0 A[3] = 103
A
def index val index val
1 0 100 3 103
(c) after A[1] := 101 low-bound(A) = 0 high-bound(A) = 3
A[0] = 100 A[1] = 101 A[2] = 0 A[3] = 103
A
def index val val index val
1 0 100 101 3 103
(d) after A[2] := 102 low-bound(A) = 0 high-bound(A) = 3
A[0] = 100 A[1] = 101 A[2] = 102 A[3] = 103
A
def index val val val val
1 0 100 101 102 103

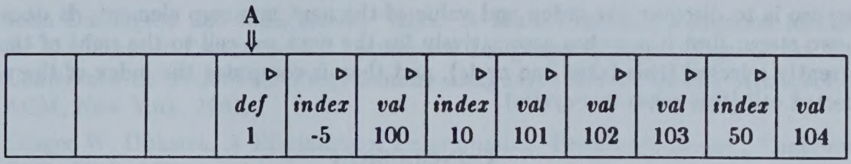
Figure 4. Examples of sparse array variable representations.

value of the index is less than or equal to 12. The effect is to set *select* in all the *index* cells that might specify the index information for $A[12]$. (The figure indicates the cells that are selected with an arrow.) Only the rightmost such *index* can be associated with the element $A[12]$, so in 5d the "select-last" instruction is used to clear *select* in the irrelevant *index* cells. The next step (5e) is to mark the *val* cells associated with the relevant index using the "mark-adjacent-values" instruction. Finally (5f) the "relative-index" instruction sets *select* in the cell that contains $A[12]$, and it also returns the contents of that cell. The relative-index instruction will only select a cell whose *mark* is already set. Thus an access to $A[13]$ would cause relative-index to clear *select* in all the cells, and the microprogram would then know that $A[13]$ is not represented explicitly — i.e., its value is 0.

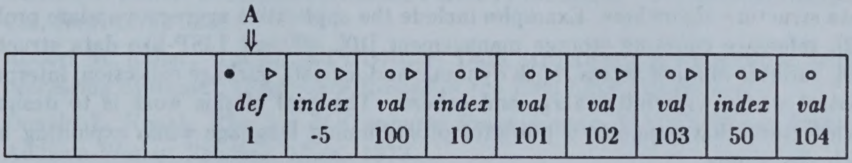
(a) initial representation

$A[-5] = 100 \quad A[i] = 0, -5 < i < 10 \quad A[10] = 101 \quad A[11] = 102 \quad A[12] = 103$

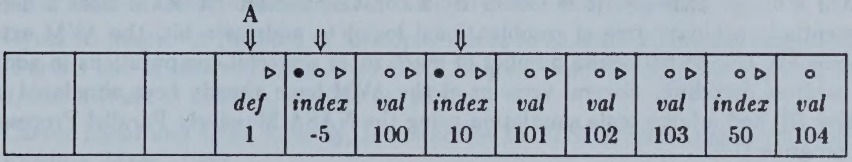
$A[i] = 0, 12 < i < 50 \quad A[50] = 104$


A
def 1
index -5
val 100
index 10
val 101
val 102
val 103
index 50
val 104

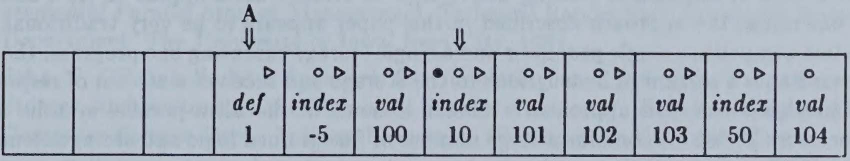
(b) select (A); mark-span;


A
def 1
index -5
val 100
index 10
val 101
val 102
val 103
index 50
val 104

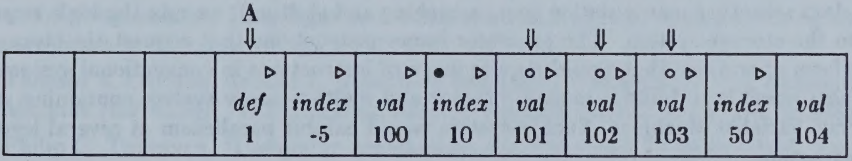
(c) select-index(12);


A
def 1
index -5
val 100
index 10
val 101
val 102
val 103
index 50
val 104

(d) select-last; returns (*index* 10)


A
def 1
index -5
val 100
index 10
val 101
val 102
val 103
index 50
val 104

(e) mark-adjacent-values;


A
def 1
index -5
val 100
index 10
val 101
val 102
val 103
index 50
val 104

(f) relative-index(12 − 10 + 1); returns (*val* 103)

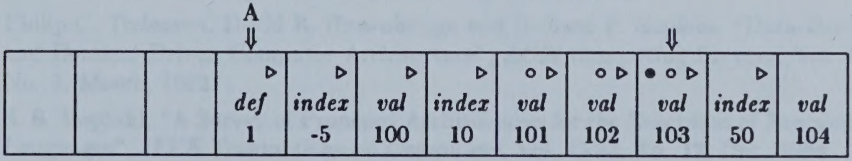

A
def 1
index -5
val 100
index 10
val 101
val 102
val 103
index 50
val 104

Figure 5. Access through direct indexing: *fetch*(*A*, 12)

The *fetch* access function, like all others, requires a constant number of cycles, regardless of the size or sparseness of the array variable. Thus the AVM satisfies the requirement that all the access functions must take a short, bounded amount of time.

The *next* access function assumes that an array element has *select* set, and its purpose is to discover the index and value of the next non-zero element. It does this in two steps: first it searches associatively for the next *val* cell to the right of the one currently selected (this takes one cycle), and then it computes the index of the newly selected cell (this takes two cycles).

CONCLUSION

The architecture described in this paper can implement a number of other useful data structure algorithms. Examples include the applicative aggregate update problem [12], reference counting storage management [10], efficient LISP-like data structures [10], variable binding access (both dynamic and lexical), garbage collection, interpreter control stacks with full funarg, and others. The goal of this work is to design an architecture that supports a powerful programming language while exploiting many such algorithms.

The asymptotic performance of the AVM storage is the same as for an ordinary RAM storage, although it is slower by a constant factor. A RAM uses a decoder (essentially a binary tree of combinational logic) to address a bit; the AVM extends the binary tree to perform a number of much more powerful computations in addition to address decoding. Several versions of the AVM have already been simulated using Daisy [7], and a large scale simulation using the NASA Massively Parallel Processor is in progress [11].

A number of related architectures are described in [5, 8, 13, 14 and 15]. General discussions of the idea of putting active logic in storage units appear in [1, 4 and 16]. In one sense, the approach described in this paper appears to be very traditional: the system comprises a single processor and a single storage executing one program; the processor issues a stream of instructions to the storage and receives a stream of responses. In another sense, this approach is similar to some of the ultra-parallel systolic array processors [9], which combine a large number of fine grained logic and storage elements. The AVM uses parallelism *within* its instructions rather than using parallelism to execute many instructions simultaneously [2]. Instead of putting several processors to work at data structure manipulation (e.g., searching and shifting), we mix the logic required into the storage system. The processor issues instructions that request the storage to perform operations that would require loops of instructions in conventional systems. It is also possible to build a multi-processor and multi-memory system containing many Array Variable Machines. Such a system would exhibit parallelism at several levels of abstraction.

REFERENCES

1. John Backus, "Can Programming be Liberated from the von Neumann Style?", *CACM* Vol. 21, No. 8, 1978.
2. John Darlington and Mike Reeve, "Alice: A Multi-Processor Reduction Machine for the Parallel Evaluation of Applicative Languages", *Proceedings of the 1981 Conference on Functional Programming Languages and Computer Architecture*, ACM, New York, 1981.
3. Edsger W. Dijkstra, *A Discipline of Programming*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1976.
4. Caxton C. Foster, *Content Addressable Parallel Processors*, Van Nostrand Reinhold Co., New York, 1976.
5. Robert M. Keller, "Divide and CONCer: Data Structuring in Applicative Multiprocessing Systems", *Conference Record of the 1980 LISP Conference.*
6. Donald E. Knuth, *The Art of Computer Programming*, Vol. 1, Second Edition, Addison-Wesley, Reading, Mass., 1973.
7. Anne T. Kohlstaedt, *The Daisy 1.0 Reference Manual*, Technical Report 119, Computer Science Department, Indiana University, Bloomington, 1981.
8. Gyula A. Mago, "A Network of Microprocessors to Execute Reduction Languages" (Parts 1 and 2), *International Journal of Computer and Information Sciences*, Vol. 8, No. 5 (Part 1) and No. 6 (Part 2), 1979.
9. Carver Mead and Lynn Conway, *Introduction to VLSI Systems*, Addison-Wesley, Reading, Mass., 1980.
10. John T. O'Donnell, *A Systolic Associative LISP Computer Architecture with Incremental Parallel Storage Management*, Technical Report 81-5, Computer Science Department, The University of Iowa, Iowa City, 1981.
11. John T. O'Donnell, "An Approach for Simulating an Applicative Programming Storage Architecture on the Massively Parallel Processor", Technical Report 179, Computer Science Dept., Indiana University, Bloomington, Sept. 1985.
12. John T. O'Donnell, "An Architecture that Efficiently Updates Associative Aggregates in Applicative Programming Languages", *1985 IFIP Symposium on Functional Programming Languages and Computer Architecture*, Springer-Verlag Lecture Notes in Computer Science 201, 1985.
13. Thomas A. Ottman, Arnold L. Rosenberg and Larry J. Stockmeyer, "A Dictionary Machine (for VLSI)", *IEEE Transactions on Computers*, Vol. C–31, No. 9, 1982.
14. Philip C. Treleaven, "Computer Architecture for Functional Programming", *Functional Programming and its Applications*, J. Darlington, P. Henderson and D. A. Turner (ed.), Cambridge University Press, Cambridge, 1982.
15. Philip C. Treleaven, David R. Brownbridge and Richard P. Hopkins, "Data-Driven and Demand-Driven Computer Architecture", *ACM Computing Surveys*, Vol. 14, No. 1, March, 1982.
16. S. R. Vegdahl, "A Survey of Proposed Architectures for the Execution of Functional Languages", *IEEE Transactions on Computers*, Vol. C-33, No. 12, Dec. 1984.

INDEX OF AUTHORS

Agarwal, V. K. 183
Al-Baiyat, S. A. 103
Allebach, J. P. 326
Anantharam, V. 693
Anderson, J. B. 859
Ansari, N. 292
Apostolico, A. 76
Attalah, M. J. 268
Bahr, R. K. 18
Bajaj, C. 510
Barmish, B. R. 819
Bayoumi, M. 388
Beck, C. L. 940
Bellegarda, J. R. 742
Berger, T. 361
Bhargava, U. K. 584
Bilardi, G. 261
Bingulac, S. 779
Blaum, M. 452
Bosch, W. R. 370
Bose, P. 158
Bovet, D. P. 898
Bovik, A. C. 304
Bovopoulos, A. D. 564
Boyd, S. 354
Brady, M. L. 674
Brown, D. J. 674
Browning, D. W. 529
Bshouty, N. H. 168
Bucklew, J. A. 18
Bugara, K. M. 101
Buzo, A. 337
Carroll, R. L. 752
Castor, K. G. 869
Chakravarty, S. 174
Chang, B.-C. 797
Chang, C. Y. 430
Chang, Y. 270
Choi, H.-A. 720
Chou, P.-K. 594
Cobb, D. 816
Conry, S. E. 67
Costello, Jr., D. J. 428
Cotsapas, N. 183
Cruz, J. R. 606,783
Dandapani, R. 202
Dasgupta, S. 411
Daum, F. E. 2
Davidson, F. 388
Dawson, D. F. 774

Dean, J. A. 496
Dehne, F. 518
Deller, Jr., J. R. 732
Delp, E. J. 292
Duel, A. 878
Dwyer, T. A. W. 144
El-Gamel, A. 233
Eom, K.-B. 314
Ephremides, A. 555
Farden, D.C. 742
Farvardin, N. 359
Filar, J. A. 776
Fliess, M. 123
Fuhrmann, D. R. 604
Fuja, T. 455
Gafni, E. 283
Gertler, J. 419
Ghosh, B. K. 795
Goodman, D. M. 584
Goodman, R. 455
Goodman, R. M. F. 452
Gopal, I. S. 688
Gouda, M. G. 914
Gray, R. M. 354
Gu, G. 113
Gubner, J. 372
Günther, O. 644
Guerra, C. 76
Gupta, L. 301
Haddad, A. H. 777
Hakimi, S. L. 720
Halverson, D. R. 45
Hambrusch, S. 270
Han, S. 676
Harpaz, Z. 280
Heegard, C. 455,878
Herro, M. A. 401,450
Hu, L. 401,450
Hu, Y.-H. 594,764
Huang, Q. 785,806
Huang, S.-H.S. 840
Humblet, P. 390
Hunt, H. B., III 174
Janko, W. H. 850
Jin, X. 85
Kain, R. Y. 193
Karamanos, N. A. 370
Karp, R. M. 1
Kashyap, R. L. 314,584
Khalil, H. 130
Khargonekar, P. P. 787
Kinney, L. L. 193
Koivo, A. J. 417,421
Koplowitz, J. 324

Kouta, M. M. 67
Krogh, B. H. 940
Krtolica, R. 779
Kuhlmann, F. 337
Kumar, P. R. S. 730
LaFanechere, A. 428
Lazar, A. A. 564
Lee, E. B. 113
Lee, S. S. 543
Levcopoulos, C. 664
Levine, J. 9
Li, V. O. K. 545
Li, X. 774
Lin, F. Y. 359,942
Lindner, D. K. 142
Lingas, A. 480
Liu, B. 574
Liu, K. 419
Liu, R. 654,785,806
Liu, Y.-C. 533
Ljung, L. 204
Lookabaugh, T. 354
Luk, T. C. 732
Luse, D. W. 622
Marberg, J. M. 283
Massey, J. L. 361
Massey, W. A. 703
McEliece, R. J. 452,465
McIntyre, M. 951
McKenna, M. 486
Mehravari, N. 525
Melhem, R. 923
Meyer, G. G. L. 243
Michel, A. N. 146
Miller, M. I. 370
Miller, R. K. 146
Mitra, D. 710
Moayeri, N. 347
Mohan, S. 324
Montgomery, R. C. 142
Morrison, J. M. 28
Mulholland, R. J. 606,783
Murgolo, F. D. 74
Nadkarni, A. P. 531
Nandy, S.K. 686
Narayan, P. 372
Narenda, K. S. 217
Neuhoff, D. L. 347
Nocedal, J. M. 606
Ntafos, S. 654
O'Donnell, J. T. 986
O'Rourke, J. 470,486
O'Sullivan, J. A. 156
Oommen, B. J. 506

Panconesi, A. 898
Pang, K. F. 233
Papadimitriou, C. H. 63
Pascoal, A. 787
Patnaik, L. M. 686
Pawagi, S. 251
Pearlstein, L. 574
Pitstick, G. M. 783
Podrazik, L. J. 243
Pollara, F. 440
Poplawski, D. A. 979
Posner, E. C. 465
Purdom, Jr., P. W. 101
Ramakrishnan, I. V. 251
Restrepo, A. 304
Rhodes, I. B. 620
Richards, D. 830
Rief, J. H. 821
Rivera, C. 337
Rodemich, E. R. 465
Root, W. L. 624
Rosberg, Z. 688
Rosier, L. E. 914
Roth, G. 508
Saberi, A. 130
Sack, J.-R. 496
Sadowsky, J. S. 53
Sahni, S. 676
Sain, M. K. 103,156
Saluja, K. K. 202
Saniuk, J. M. 620
Sauer, K. D. 326
Schnitzer, M. S. 45
Schultz, T. A. 776
Sennott, L. I. 527
Serakos, D. 624
Seroussi, G. 168
Seshadri, N. 859
Sheen, P. 202
Silvester, J. A. 543
Simon, J. 270
Sinha, A. S. C. 618
Sira-Ramirez, H. 140
Snyder, D. L. 336
Srinath, M. D. 301
Sriraman, R. 324
Stark, W. E. 347,869
Stigall, P.D. 774
Su, S.-L. 545
Sugano, J. 971
Sullivan, B. J. 616
Sung, H.-L. 764,906
Sung, T.-Y. 764
Suri, S. 470,486

Thistle, J. G. 961
Thomas, J. B. 35,529,888
Tong, Z. 193
Tsitsiklis, J. N. 63
Ueda, K. 971
Vastola, K. S. 531
Vaz, A. F. 933
Verdu, S. 382
Verriest, E. I. 777
Vidyasagar, M. 951
Vidyasankar, K. 92
Viniotis, I. 555
Walrand, J. 708
Warren, D. 35
Willard, D. E. 821
Willett, P. K. 888
Wise, G. L. 28,533
Wonham, W. M. 933,942,961
Wu, Y. A. 762
Yao, K. 430
Yaz, E. 781
Yen, H.-C. 914
Young, G. K. 730
Yu, H.-J. 764
Zaks, S. 280
Zarei, S. 417
Zhang, S.-Y. 634
Zhang, Z. 361
Zilouchian, A. 752

UNIVERSITY OF ILLINOIS-URBANA
621.3811AL5 C001
PROCEEDINGS / ALLERTON CONFERENCE ON COM
1985
3 0112 008149202